AFRL-IF-RS-TR-2005-332
Final Technical Report
September 2005

CYBER SIGNAL/NOISE CHARACTERISTICS AND SENSOR MODELS FOR EARLY CYBER INDICATIONS AND WARNING

Arizona State University

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-332 has been reviewed and is approved for publication

APPROVED: /s/

WLADIMIR TIRENIN
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	SEPTEMBER 2005	Final Sep 03 – Mar 05

4. TITLE AND SUBTITLE
CYBER SIGNAL/NOISE CHARACTERISTICS AND SENSOR MODELS FOR EARLY CYBER INDICATIONS AND WARNING

5. FUNDING NUMBERS
C - F30602-03-C-0233
PE - 31011G
PR - B104
TA - 00
WU - 03

6. AUTHOR(S)
Nong Ye

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Arizona State University
Industrial Engineering Department
Tempe Arizona 85287-5906

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFGB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-332

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Wladimir Tirenin/IFGB/(315) 330-1871/ Wladimir.Tirenin@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
We designed a method to develop a suite of specialized cyber sensors that are optimized to detect cyber attack observables. We develop our sensors using scientific knowledge of characteristics of cyber signal (attack data) and noise (normal "norm" data). In our approach, we built models for attack norm characteristics. To detect characteristics, we used our norm model to filter out noise from mixed data and our attack model to detect a cyber signal. Our solution aims to reduce false alarm rates, increase detection rates and provide earlier detection with knowledge gained from our scientific investigation of attacks.
The development phases of the attack-norm separation approach include classifying and profiling cyber attacks, analytical discovery of signet and noise characteristics, designing and testing sensor models, sensor fusion models, and finally an optimized suite of cyber sensors. We have created a number of sensors based on a subset of cyber attacks and tested them to show performance of attack detection and recognition.

14. SUBJECT TERMS
Cyber Signal, Cyber Sensors, Cyber Attack, Attack Data, Attack Noise, Sensor Fusion Models, Attack Detection, Sensors

15. NUMBER OF PAGES
195

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

Summary

We designed a method to develop a suite of specialized cyber sensors that are optimized to detect cyber attack observables. We develop our sensors using scientific knowledge of characteristics of cyber signal (attack data) and noise (normal “norm” data). In our approach, we built models for attack norm characteristics. To detect characteristics, we used our norm model to filter out noise from mixed data and our attack model to detect a cyber signal. Our solution aims to reduce false alarm rates, increase detection rates and provide earlier detection with knowledge gained from our scientific investigation of attacks.

The development phases of the attack-norm separation approach include classifying and profiling cyber attacks, analytical discovery of signal and noise characteristics, designing and testing sensor models, sensor fusion models, and finally an optimized suite of cyber sensors. We have created a number of sensors based on a subset of cyber attacks and tested them to show performance of attack detection and recognition.

1. Introduction

A suite of specialized cyber sensors, that are optimized to detect cyber attack observables, is imperative because of the efficiency, accuracy, and adequacy problems encountered by existing intrusion detection systems that use event streams (raw activity data). As we see how physical sensor technologies (e.g., radar sensors, sound sensors, light sensors, etc.) have been developed for signal detection in the physical world, the development of sensors for cyber signal detection will need to rely on the scientific knowledge of characteristics of cyber signal (attack data) and noise (normal "norm" data) to achieve detection efficiency, accuracy and adequacy. However, characterization of cyber attack and normal data is far from fully established, and little scientific knowledge exists. We designed a method for building a sensor grid based on scientific knowledge of attack and normal data and developed an optimized suite of cyber sensors using:

- An innovative combination of risk assessment, system modeling, and fault modeling theories that capture resource-process, activity-state-performance, and attack-threat-mission interactions temporally, spatially, and functionally in cause-effect networks of attack profiles
- Scientific discovery of cyber signal and noise characteristics
- Quantitative and qualitative models for cyber attack detection and recognition based on signal processing, detection, and time series analysis theories, cu-score statistic test techniques, and other technologies, according to characteristics of cyber signal and noise
- Sensor fusion models based on decision and fusion theories
- Optimization of I&W observables/cyber sensors based on Operations Research theory

The following sections in this report summarize our research. We start by describing existing intrusion detection systems and their shortcomings. We then introduce our attack-norm separation approach. Our approach involves a number of steps which we describe in the subsequent sections on attack classification and profiling, characteristics of cyber signal and noise, sensor and sensor fusion models, and optimized suite of cyber sensors. Following these sections, we present the results of our two subcontractors on this project: Symantec and AT&T. Finally, we conclude this report.

2. A New Approach to Cyber Attack Detection & Recognition

We first describe two existing approaches to intrusion detection: signature recognition and anomaly detection. We then present a new approach, called attack-norm separation, and compare it with the two existing approaches. (Note that in the earlier months of this project, we were calling our approach signal-noise separation. This caused some confusion and we changed it to attack-norm separation. Thus, where previous reports used the terms signal and noise, we use attack and norm in this report.)

2.1 Existing Intrusion Detection Techniques

Two existing approaches to intrusion detection are signature recognition and anomaly detection. Most commercial intrusion detection systems, including anti-virus software, employ signature recognition to detect cyber attacks. In this approach, signature patterns of attacks are either manually captured by expert analysts or automatically discovered through mining computer and network activity data collected under attack and normal operating conditions [1,2]. Attack signatures are stored and used in an intrusion detection system to check against activities and files on computers or networks for the presence of a signature. If present, the system detects an attack. Since the signature patterns of novel attacks are often unknown, signature recognition is not effective against them.

Anomaly detection considers any large deviation from normal system behavior as an indication of a possible attack [2,3]. Thus, it requires an established model of normal system behavior (norm profile), to monitor activities on computers and networks and measure deviations from the norm. A large deviation indicates a possible attack. We can establish a norm profile according to the system norm by design, or by learning from data of system behavior collected under normal operating conditions. Various norm profile modeling techniques have been investigated, including strings representing sequences of system calls, Statistical Process Control (SPC) charts, Markov chain models, data clusters, association rules and artificial neural networks [2,3]. An anomaly detection technique can detect a novel attack if it shows a large deviation from its norm profile. However, a novel attack may not deviate largely from the norm profile, yielding a miss or detection failure. Furthermore, the modeling technique used in an anomaly detection solution may not be powerful enough to cover all kinds of normal system behavior, especially that which is normal, but irregular. When such behavior occurs, the solution erroneously indicates a possible attack, yielding a false alarm. Too many false alarms burden system administrators, who must investigate them, rendering the anomaly detection approach impractical to some extent. Hence, in spite of its advantage in possibly detecting novel attacks, anomaly detection has not become popular in commercial intrusion detection systems.

Essentially, both approaches employ data analysis combined with a model of system behavior to detect attacks. The two approaches differ in their underlying models. Signature recognition uses a model of “bad” system behavior under the attack condition, whereas anomaly detection uses a model of "good" system behavior under the normal operation condition. Attacks on computers and networks are detected when the observed behavior either correlates with known attack profiles or diverges from known normal profiles. Neither of the two approaches requires and enforces the use of both attack and normal behavior models in contrast to achieve detection accuracy. The mixture of attack and norm data extracted for intrusion detection weakens the distinctive characteristics of both, resulting in poor detection performance (including misses and false alarms). If we consider a data characteristic to be strong or weak, there are four combinations of characteristics in mixed data shown in Table 1.

Table 1: Characteristics of attack and norm data

		Attack Characteristic	
		Weak	*Strong*
Norm Characteristic	*Weak*	weak norm, weak attack	weak norm, strong attack
	Strong	strong norm, weak attack	strong norm, strong attack

Existing techniques work well for only one combination: strong attack and weak norm characteristic. A signal detection model, incorporating characteristics of both signal and noise mixed together in monitored data, can more accurately detect a signal in noise than a model relying on only one element, and is more sensitive to low signal-to-noise ratios (where the signal is buried in a lot of noise) [4]. A low signal-to-noise ratio is often the case in cyber attack detection since there are usually many more normal activities than attacks on computer and network systems.

2.2 Attack-Norm Separation Approach

In our approach, we built models for cyber signal (attack) and noise (norm) characteristics. To detect characteristics, we used our norm model to filter out noise from mixed data and our attack model to detect a cyber signal. Table 2 illustrates how attack-norm separation differs from signature recognition and anomaly detection.

Table 2. Comparing intrusion detection techniques

	Anomaly Detection	**Signature Recognition**	**Attack-Norm Separation**
Detect deviation from normal (possible attack)	X		X
Identify known attack		X	X

As shown in Table 2, our approach detects norm deviations, which could indicate a possible attack, and identifies the attack if it is known. For novel attacks, our solution draws on generalized information from profiles of known attacks to classify observed anomalies into possible attack categories. We aim to reduce false alarm rates, increase detection rates and provide earlier detection with knowledge gained from our scientific investigation of attacks.

Unlike current solutions, which monitor only activity data, the attack-norm separation approach considers the true normal space, to contain activity, state and performance data, thus providing adequate coverage of the cause-effect propagation data space associated with normal user activities and attacks. Each sensor model calls for the monitoring and processing of only a small amount of specific data to provide certain characteristics of attack and norm activity. Hence, each model is efficient, accurate, and adequate in detecting a given attack signal in normal noise. This approach aims to raise the level of detection accuracy, reduce the amount of

monitored data, improve the relevance of monitored data to intrusion detection, and allow for easy protection of a small amount of specific data.

Figure 1 shows the steps of our development process. First, we classified several cyber attacks and discovered their associated cause-effect networks to identify observable points for cyber attack detection. Next, we characterized observable points using signal processing, time series, and other models of signal detection. The first two steps form an iterative process. As we explored the characteristics discovered through analysis, we verified them with the attack profiles. This focused verification gave us a two-way validation of the expected observables from the profile, and observations made in the analysis.

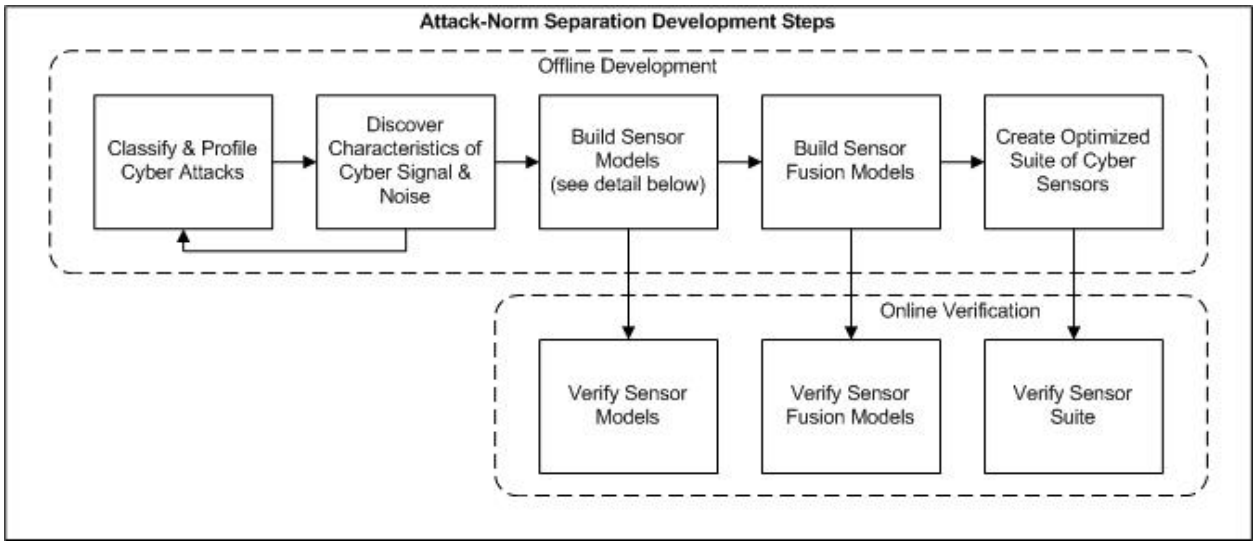


Figure 1. Method for developing an optimized suite of cyber sensors

The third and forth steps in Figure 1 are to develop sensor and sensor fusion models, which fuse decision outcomes from different sensors for the same cyber signal and noise data into an integrated decision. We started by taking results from the first two steps and developing sensor models with collected simulation data (offline development). We then verify the models in real-time (online verification). This verification confirms that the sensors are working as expected. Finally, we developed an optimized suite of cyber sensors that uses the least number of sensors necessary to accurately detect and identify the attack profiles in our study.

We expect that attack coverage for our attack-norm separation approach will expand with increasing knowledge of attack and norm characteristics, just as signal detection knowledge and technologies in the physical world evolved. A comprehensive knowledge of cyber attack and norm characteristics establishes a solid, scientific foundation of cyber attack detection to help overcome the shortcomings of existing empirical techniques.

3. Attack Classification and Profiling

The first step in building sensors for our sensor grid is attack classification and profiling. In order to discover characteristics of attacks, we need to first have a better understanding of the attacks considered in our study. We have looked at over 100 attacks, and classified a number of them using the scheme presented in this section.

Rigorously cataloguing computer and network attacks involves two steps: classification and profiling. In attack classification, we organize attacks into a classification table and tree based on the nature of the attack. In attack profiling, we outline the steps involved in the setup and execution of an attack. With these tools, we can identify the pre-attack phases of an attack and the observable points that make up the attack's signature. Identifying the observable points in an attack enables the design of a sensor model to detect it. Additionally, the sensor model can be designed to specifically identify an attack in its pre-attack phase, thereby allowing room for measures to counter the attack before the strike. This is much more desirable than detecting an attack after it has begun to propagate and cause damage.

In this section, we give background material for the design of our attack classification tree and profiling method. The following subsections present attack classification and attack profiling. Finally, we conclude this section.

3.1 Background

We categorize attacks using three theories to provide a scientific foundation: risk assessment, system modeling and fault modeling.

3.1.1 Risk assessment theory

Three factors contribute to any risk: asset, vulnerability, and threat [5,6]. We define each of these terms and use them in our attack classification scheme.

For the risk of cyber attack, assets are what the defender needs to protect on computers and networks. Assets include information processing, storage and communication resources such as CPU and memory at the hardware level, and the operating system, data files, databases, application programs, and network programs at the software level. Each asset is assigned with an asset value to measure the relative importance of the asset in the defender's missions. A mission model can be constructed to specify missions and their projection onto assets in the defender's information infrastructure. The asset value can be derived from the projection of missions onto a given asset. Three attributes of an asset have security impact on the asset value: availability, integrity and confidentiality. The availability attribute of an asset describes whether or not users with access rights to the asset can access the asset and obtain service at any time when service is needed. The confidentiality attribute of an asset describes whether or not the content or operation of the asset is kept secret from unauthorized users. The integrity attribute of an asset describes whether or not the content or operation of the asset can be kept accurate without unauthorized alteration or deletion, and thus service from the asset can be trusted. A cyber attack that causes a

change from the desired level of the availability, confidentiality and integrity state of an asset results in a compromised asset value, and becomes a security problem.

Vulnerability evaluates the security strength of an asset. An asset is vulnerable if there is an opportunity to cause the damage or loss of the asset value. An asset may have more than one vulnerability. A vulnerability value can be assigned to indicate the severity of asset damage or loss if the vulnerability is exploited in a malicious cyber attack.

While assets are what we are protecting, threats are potential attacks that we are protecting from. A threat value can be assigned to indicate the likelihood or potential of the threat. The attacker's threat profile can be established to characterize the attacker's sources (e.g., nation states, terrorists, criminal elements, hackers, or corporate competitors, etc.), attack capabilities (e.g., resources, skills, tools, methods including passive, active, close-in, insider, and distribution, etc.), motivations (e.g., malicious versus non-malicious, intelligence gathering, theft of intellectual property, causing embarrassment, pride and proof of skills, etc.), status (e.g., outsider, insider, etc.), readiness (e.g., how much intelligence information about the target system the attacker has possessed), and so on.

There is no risk if any one of the three contributing factors—asset, vulnerability, or threat—does not exist. An asset may have multiple vulnerabilities, each of which may be subject to multiple applicable threats.

3.1.2 System modeling theory

A system consists of two basic elements: resource and process [6,7]. A resource in the system, corresponding to an asset in risk assessment theory, provides service to a process requesting service from the resource. Multiple processes may request service from the resource at the same or different times.

A process has input and output. Servicing a process changes the state of a resource. A change in the resource state in turn has impact on the output performance of the process. For information security, there are three attributes of resource state: availability, confidentiality, and integrity as defined in risk assessment theory, and there are three attributes of output performance: timeliness, precision, and accuracy. Timeliness measures how long it takes to produce the output. Precision measures how much output is produced, related to the quantity of the output. Accuracy measures the correctness of the output, related to the quality of the output. The availability, confidentiality, and integrity attributes of the resource state affect the timeliness, precision, and accuracy attributes of output performance respectively. For example, a CPU is a resource or an information processing asset. When a CPU services a process, the availability attribute of the CPU state changes because less CPU time becomes available. The availability state of the CPU in turn affects the timeliness attribute of the output performance for the process. Activities in the system include user activities to initiate processes and receive service and operations of resources to provide service. User activities to initiate processes change the state of resources, and changes in the resource state in turn have impact on the output performance of processes. Hence, in a system consisting of resources and processes, there are resource-process interactions and activity-state-performance interactions.

3.1.3 Fault modeling theory

A fault has a propagation effect in a system, involving activity-state-performance interactions as discussed in system modeling theory [7]. Hence, a fault can be modeled in a cause-effect chain or network of activity, state change, and performance impact, all occurring in the system during the fault effect propagation. For information security, the attacker's activities cause the state change of resources on computers and networks, which in turn produces performance impact (e.g., performance error of processes or degraded quality of service provided by resources to processes), as shown in Figure 2.

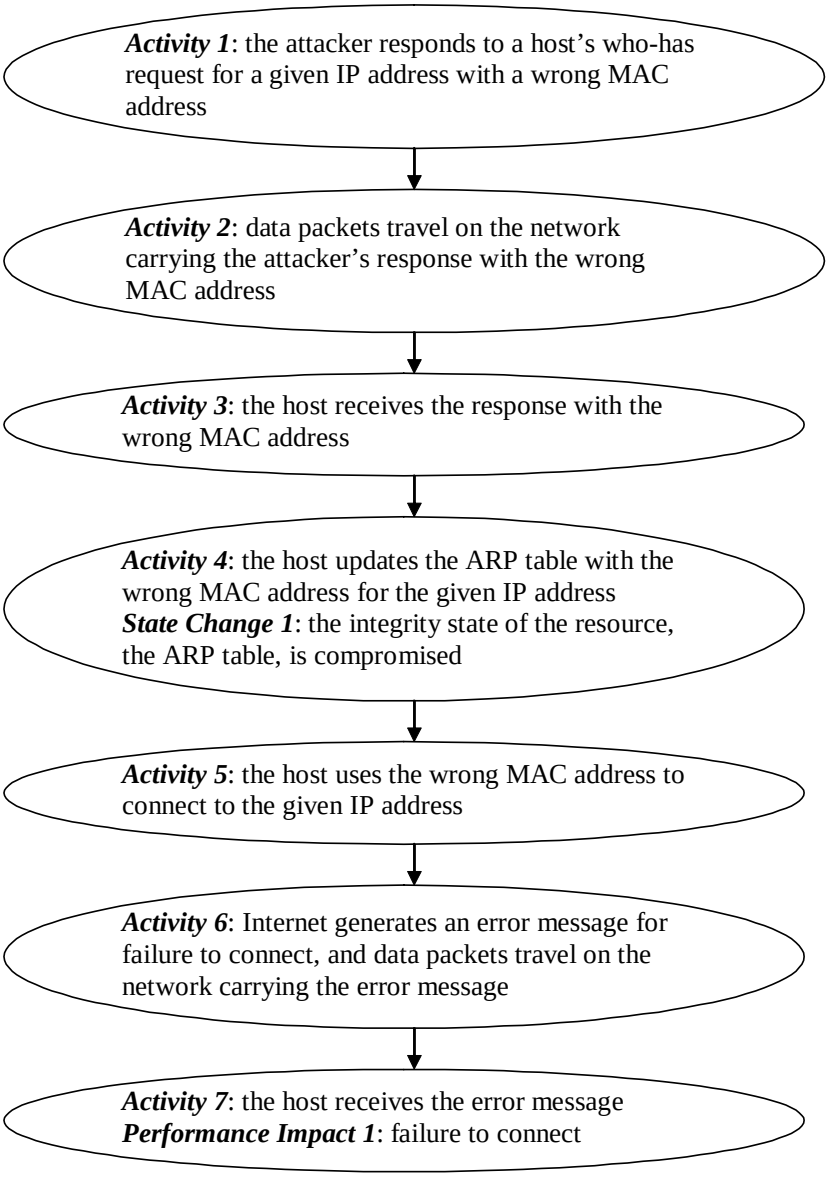


Figure 2. The cause-effect chain of an ARP poison attack.

3.1.4 Intersection of theories

Although the above theories have been applied separately for various aspects of information security and assurance, they have never been combined to predict cyber attacks. We propose to innovatively combine the above theories in our proposed solution for the sensor grid (see Figure 3).

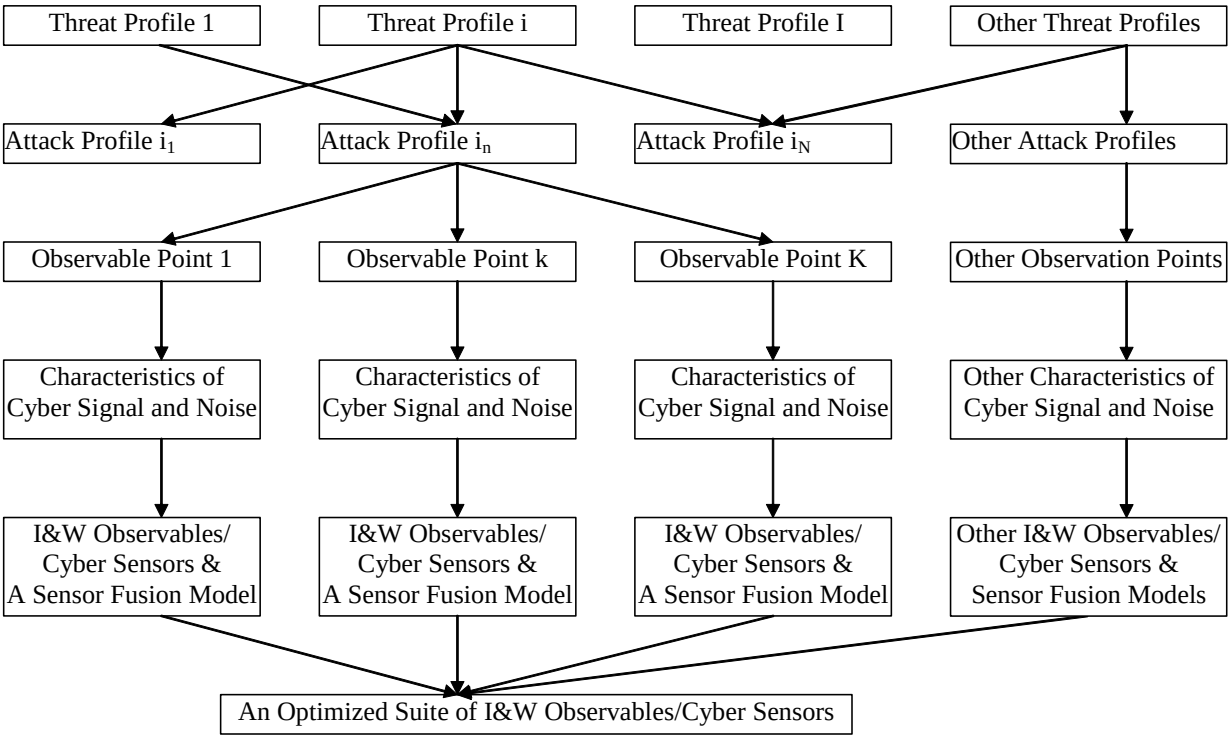


Figure 3. The proposed solution for the sensor grid.

Our proposed solution for the sensor grid will include threat profiles, attack profiles, observable points and characteristics of cyber signal and noise at those points. These concepts are described below.

A threat profile captures the threat factor in risk assessment theory. Threat profiles represent various types of cyber threats. Such profiles may consist of information that describes the nature of a threat, such as an attacker's source, capabilities, motivation, status, readiness, etc.

Attack profiles exist for each threat profile. An attack profile takes the form of a cause-effect chain or network of activities, state changes, and performance impacts with resource-process interactions and activity-state-performance interactions throughout the course of an attack scenario, as the one shown in Figure 2, but also enlarged with elements from a threat profile to present a comprehensive picture of an attack scenario under a threat profile. Under each threat profile, a number of attack scenarios may be applicable, and therefore there will a

number of attack profiles, one for each attack scenario. The same attack profile may be used under different threat profiles.

When we consider an attack profile under a threat profile, elements of the threat profile will be added to the cause-effect or network of the attack profile. For example, elements about the nature of the attack (i.e. attacker capabilities) in the threat profile may be added to the cause-effect chain or network of the attack profile as pre-conditions for certain activities, state changes, or performance impacts along the cause-effect chain or network. After being enlarged with elements of the threat profile, the attack profile will become a cause-effect network if it is a cause-effect chain before the enlargement.

Therefore, the cause-effect network of the attack profile, enlarged with elements of a threat profile, will combine the asset, vulnerability and threat factors in risk assessment theory, and combine risk assessment theory with system modeling and fault modeling theories. Incorporating elements of the threat profile into the cause-effect network of the attack profile will improve detection efficiency and accuracy, and increase the warning time for early cyber I&W.

Nodes in the cause-effect network of an attack profile will represent the observable points, such as activities (or events), state changes, and performance impacts. Directed links between nodes will represent cause-effect relationships between nodes. Observable points along the cause-effect network of each attack profile. Each node in the cause-effect network of each attack profile will become a candidate observable point. Only those observable points that have cyber sensors from an optimized suite of cyber sensors (discussed below) will be selected to be observable points that will be monitored by cyber sensors in the sensor grid.

Characteristics of cyber signal and noise at each observable point that will be important in cyber signal detection, including:

a) Statistical characteristics, such as mean, variance, probability distribution, covariance, auto-correlation, dependency, and stationarity.

b) Characteristics of spatial and temporal correlations, such as frequency band, shift, trend (i.e., cyclic and seasonal), drift (i.e., upward and downward), intermittent spike or bump, and change (step change, exponential change, slope change, sine wave, square wave, etc.), characteristic changes in dynamic state, phase synchronization, etc.

Using the concepts outlined in this section, we create an attack classification scheme to capture the threat profile along with aspects of the attack profile in a table and tree format to begin cataloguing computer and network attacks.

3.2 Attack Classification

We develop a system fault risk (SFR) framework for cyber attack classification based on the theories described in the previous section. Because many attacks have different forms, we must understand the similarities and differences between attacks. Our classification framework simplifies the task of comparing attacks. This framework allows us to group attacks to develop intrusion detection techniques based on group characteristics. In this section we describe the SFR classification framework for classifying cyber attacks. The SFR framework incorporates a cause-

effect chain into its design. To build the classification structure, we first consider vulnerabilities in computer and network systems.

Vulnerabilities can be created in otherwise secure systems by improper configuration of a system or the installed software. For example, certain systems and software packages ship with a default account and password. A common configuration error is forgetting to change or remove that default account, and this leaves a wide-open back door for attackers. Implementation errors can introduce vulnerabilities into systems that have perfectly secure implementation and design. Buffer overflow vulnerabilities caused by improper bounds checking on variables is a common example of this error. Specifications may contain weaknesses by design or error, and these weaknesses cannot be corrected later in design or implementation. For example, consider the TCP Reset attack where the attacker listens for connections to a victim computer. When a client attempts to connect to the victim, the attacker sees it and sends a TCP reset packet to the victim that is spoofed to appear to have come from the client. In this way the client uses the TCP specification to tear down any attempted connections to the victim. Human gullibility is a constant source of vulnerability as attackers routinely attempt to fool people into revealing critical information such as usernames, passwords, and credit card numbers.

Using the SFR framework, and taking into account vulnerabilities present in computer and network systems, we build an attack classification table in the next section. For another perspective, we also create attack classification trees in the following section.

3.2.1 Classification Tables

SFR and vulnerability analyses combined with in depth conceptual analysis of individual attacks produce a collection of factors involved in cyber-attacks. The factors revealed are sorted into the categories: objective, propagation, attack origin, action, vulnerability, asset, state effects and performance effects. These categories make up an incident. The incident encompasses the two subclasses: threat and attack, and a cause-effect chain. This attack classification scheme is shown in Figure 4.

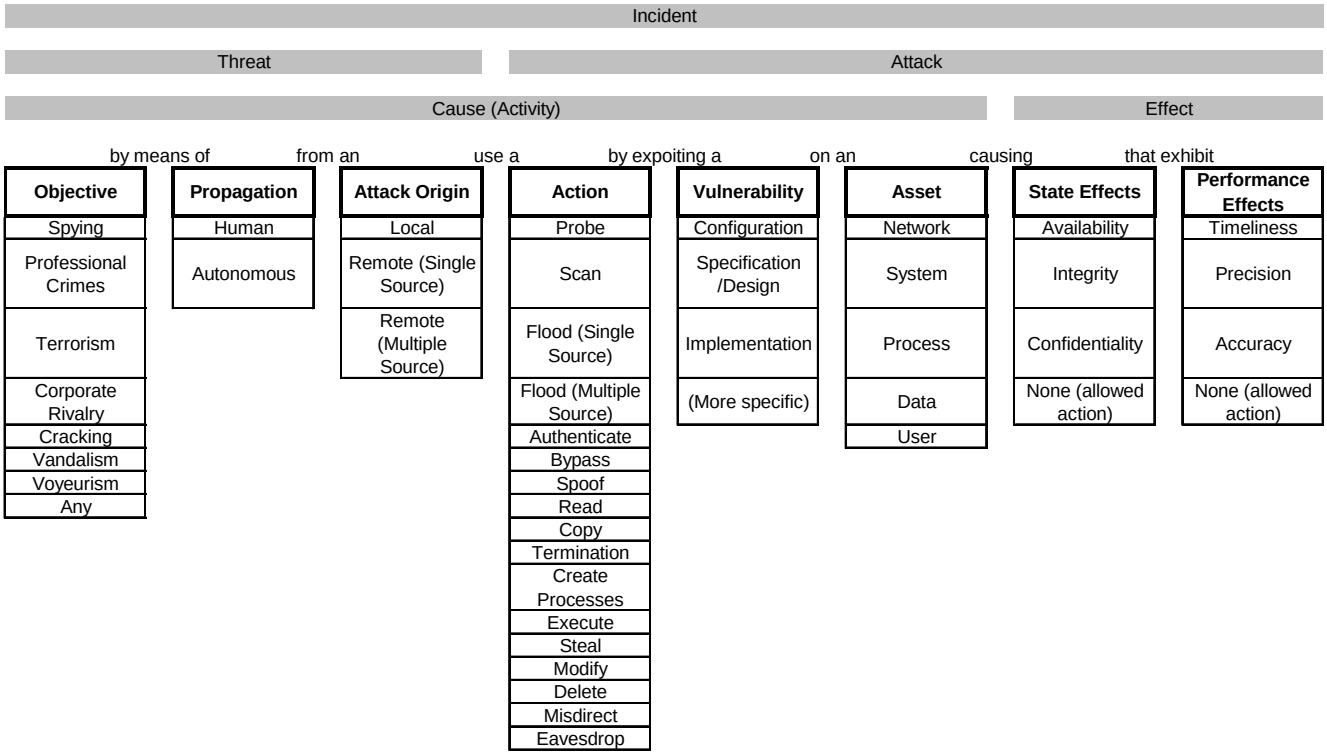


Figure 4. Framework of cyber attack classification

We describe the purpose of each column shown in Figure 4 and outline the definitions of the elements present in each column:

- Objective: This column is a combined description of technical skills, resource levels, and potential threat sources (which implies intent).
  - Spying: This can occur at a government or corporate level. In either case, the perpetrator(s) are assumed to have high skills and potentially high levels of resources at their disposal. This type of attack is politically motivated and includes interests such as information theft and sowing disinformation.
  - Professional Crimes: Committed by perpetrator(s) whose objectives are financially based. This could be an individual or organization. Potentially high skill level and resources.
  - Terrorism: This can be conducted by perpetrator(s) of varying, but potentially high, skill levels. Terrorists are politically motivated and are interested in information theft, and destruction of resources on a massive scale.
  - Corporate Rivalry: An organization with limited resources and a high technical ability. Corporations are interested in information theft to gain an advantage on their competition. They may also be destructive and intend to harm competitive corporations.
  - Cracking: A single individual (cracker) likely with limited resources and a high technical ability. Usually motivated by a technical challenge and may or may not be destructive.
  - Vandalism: A single individual with few resources and modest technical ability. Vandals want to make their mark on the world, and do so by defacing or destroying assets.

- Voyeurism: A single individual with few resources and modest technical ability. Voyeurs are curiosity seekers and typically any damage they cause is usually accidental.

- Propagation: This column is used to describe the level of propagation employed by the attack. For our purposes it is sufficient to distinguish between human controlled propagation of attacks and the autonomous propagation of viruses and worms.
  - Human: Human propagation attacks are those actively controlled by a human being. This includes Trojans that provide a back door to the system, scheduled distributed DoS attacks, and most other attacks.
  - Autonomous: Those attacks executed by self-perpetuating automated processes. Little or no human interaction is needed for these attacks to be successful. When human interaction is required, it is on the part of an unsuspecting user. This class of attacks includes viruses and worms. Trojans containing viruses and worms are also included in this category.
- Attack Origin: This column describes the physical origin of the attack with respect to the victim machine.
  - Local: This is an attack that is initiated on the machine being attacked. An example of a local attack is a user logged onto a machine who then attempts to gain root access to the same machine. (This is an insider attack).
  - Remote (single source): This is an attack that originates somewhere other than on the machine being attacked, and has a single point of origin. An example of a remote single source attack would be an unauthorized user attempting to gain access to a system over a network. (This is an outsider attack).
  - Remote (multiple sources): This is an attack that originates somewhere other than on the machine being attacked, and has multiple points of origin. An example of a remote multiple source attack would be any distributed denial of service attack.
- Action: This column descries what specific activity the attacker is performing on the victim.
  - Probe: “Access an asset in order to determine its characteristics” [8].
  - Scan: “Access a set of assets sequentially in order to identify which assets have a specific characteristic” [8].
  - Flood (Single Source): “Access an asset repeatedly in order to overload the asset’s capacity” [8]. In this case, all of the flooding data comes from a single location.
  - Flood (Multiple Source): “Access an asset repeatedly in order to overload the asset’s capacity” [8]. In this case, the flooding data comes from 2 or more locations.
  - Authenticate: “Present an identity of someone to a process and, if required, verify that identity, in order to access an asset” [8].
  - Bypass: “Avoid a process by using an alternative method to access an asset” [8]. This can be used to gain access to, or elevate privileges in a system.
  - Spoof: “Masquerade by assuming the appearance of a different entity in network communications” [8].
  - Read: “Obtain the content of data in a storage device, or other data medium” [8]. This activity implies that open, read, and possibly close operations are performed on a static file or data source.
  - Copy: “Reproduce an asset leaving the original asset unchanged” [8].

- Termination: Terminate a running process. This is frequently done in single point denial of service attacks, where an attacker will use a buffer overflow to kill a process.
- Create Processes: Spawn multiple processes, child or otherwise. This action is used by denial of service attacks that attempt to fill up the process table on a system.
- Execute: Execute as a process on a system. Typical of viruses and trojans, this is usually part of a multiple step attack where code is executed on the victim machine.
- Steal: "Take possession of an asset without leaving a copy in the original location" [8].
- Modify: "Change the content or characteristics of an asset" [8].
- Delete: "Remove an asset, or render it irretrievable" [8].
- Misdirect: Literally, "To lead in the wrong direction". In this case the act of misdirection is deliberate and deceitful. Such as, fulfilling a request to an asset that appears to be legitimate, but in actuality is a subterfuge used to extract information from the asset. Cross site scripting is an example of this. Since we include the concept of deceit, misdirection covers any attempt to lie to an asset and provoke an action based on the falsehood. Thus, email scams are counted as misdirection.
- Eavesdrop: The extraction of data from a dynamic and transient data stream. This activity implies that the collection process does not significantly disturb the data stream.

- Vulnerability: This column describes the type of vulnerability that is being exploited by the attacker. In the categorization we list the primary sources of vulnerabilities, which are sufficient for our purposes.
  - Configuration: This vulnerability occurs when a resource is configured improperly, and as a result a security hole is created [8]. An example of this could be any system or software that ships with a default account that is not changed or removed upon setup.
  - Specification/Design: "A vulnerability inherent in the design or specification of hardware or software whereby even a perfect implementation will result in a vulnerability" [8].
  - Implementation: "a vulnerability resulting from an error made in the software or hardware implementation of a satisfactory design" [8].
- Asset: This column describes the component that is under attack. Again, while this is a fairly high level list, it suits our purposes well.
  - Network: "An interconnected or interrelated group of host computers, switching elements, and interconnecting branches" [8].
  - System: "A device that consists of one or more associated components, including processing units and peripheral units, that is controlled by internally stored programs, and that can perform substantial computations, including numerous arithmetic operations, or logic operations, without human intervention during execution. Note: May be stand alone, or may consist of several interconnected units" [8].
  - Process: "A program in execution, consisting of the executable program, the program's data and stack, its program counter, stack pointer and other registers, and all other information needed to execute the program" [8].
  - Data: Representations of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or automatic means [9]. Data can be in the form of files in a computer's volatile or non-volatile memory, or a data storage device, or in the form of data in transit across a transmission medium [8].

- User: A user has at least some access privileges on a specified system. These access rights may vary from user to user. Attacks against users are typically attempts to mislead or misdirect. Examples of user attacks include misleading emails (scams) and misdirection through cross site scripting.

- State Effect: This column is used to describe the state change that occurs on the victim as a result of an attack.
  - Availability: The availability state of a resource is related to the responsiveness of the resource in meeting service requests [8]. A successful attack on availability will cause the response time of a request to an affected service to increase, or the service to become entirely unavailable. By definition, all denial of service attacks exhibit availability effects. An example would be the UDP storm attack, where availability of the UDP echo service to legitimate requests is reduced on the victim machine.
  - Integrity: The integrity state of a resource is related to the correctness of a resource in meeting service requests [8]. Integrity also includes the concept of data validity. No user (authorized or not) should intentionally or accidentally corrupt data, and unauthorized users should not modify data in any way. It follows from this that anything an unauthorized user does that alters or adds a file, user account, changes permissions, or modifies data on the system in any manner is an integrity violation. An example of an attack that affects integrity is the ARP poison attack. In the ARP attack an unauthorized user places invalid MAC addresses into the victim machines ARP table.
  - Confidentiality: The confidentiality state of a resource is related to the precision of the resource in meeting service requests, that is, "whether the resource produces the precise amount of output for a given input" [8]. A confidentiality attack aims at producing more information than normal for a given request. For example, an attacker who is monitoring a network transmission will collect a duplicate copy of a transaction, with the result that twice as much data will be produced as normal, one copy for the recipient, and one for the attacker. A successful attack on confidentiality could provide the attacker with information about the network, individual systems, system processes, and or data on a host. Furthermore, since use of resources is part of a trust relationship, (specified or not) unauthorized use of resources should result in zero output. Thus any successful unauthorized use of resources can also be considered a confidentiality violation. Note that under this definition, if an intruder were to break into a system and use the printer, all of the services used including the printer would experience confidentiality violations. A classic example of a confidentiality attack is the ping sweep.
  - None (allowed action): This represents no state effect. Some attacks begin with a step that is a legal user activity. In this event, that step has no detectable state change.
- Performance Effect: This column is used to describe the performance change that occurs on the victim as a result of an attack.
  - Timeliness: A measure of how fast an output is generated based on a given input [8].
  - Precision: This is how much output is produced for a given input [8]. The amount of output should always be 100% of the expected output. Certain conditions can cause values of other than 100% output. For example, the resource being used could crash in the middle of an operation, resulting in only a small portion of the expected output being produced. As a further example, if an employer is using a key logger, than every

keystroke produces at least 200% of the expected output by the user, one copy of the keystroke for the application, and one copy for the logger.

- Accuracy: A measure of how good an output is when related to the content quality of the expected output [8]. One method of measuring this is through a distance value between a computed checksum for the current data stream and the original (expected) data stream [8]. An example of an attack effecting accuracy would be a man in the middle attack, where the packet MAC and/or IP addresses are modified.
- None (allowed action): This represents no performance effect. Some attacks begin with a step that is a legal user activity. In this event, the step has no detectable performance change.

In keeping with fault modeling theory, our classification is ordered in terms of cause and effect. The overall incident begins with a threat, followed by an attack. A threat is composed of the three columns: objective, propagation and attack origin the attack includes action, vulnerability, asset, and state and performance effects. Additionally, the first six columns indicate the cause, while the last two the threat.

This classification scheme takes the form of checklist taxonomy and largely supports and extends the work done by Howard and Longstaff [9]. It is easy to understand, modify and extend. The relationship between cause and effect within an attack is clear. Beyond that, the relationship of the threat and attack within the overall incident is clear as well. Some of the columns in the classification scheme are very high level, and could be broken down into more elements, or even sub-lists. For our purposes of separating cyber-attacks into domains, the high level categorizations are sufficient.

It is important to note that in this format, not all possible combinations of factors make sense. Obviously, for any attack that is being classified, combinations of factors that make sense and describe the attack should be selected. There are at least two ways in which to employ this system to classify attacks. The first is to traverse all of the lists and iteratively select all the factors from each category that are descriptive of the attack. Using this method produces results that are unsuitable for taxonomy, but are sufficient for work involving ontology. The second method is to refine the attack into its constituent parts such that each part possesses only a single entry from each of the lists. This method does produce results that can be put into taxonomy.

We give four example attacks to show how attacks are classified using our classification system in Table 3. Note how the classification factors extracted from each example can be tabulated for comparison. We give descriptions of each of these four attacks below, along with explanations regarding their classification in Table 3.

Table 3. Example of cyber-attack classifications in table format

Attack Name	Objective	Propagation	Attack Origin	Action	Vulnerability	Asset	State Effect	Perf Effect
UDP Storm	Any	Human	Remote (Single Source)	Flood	Specification / Design	Network	Availability	Timeliness
Slammer Worm	Cracking	Autonomous	Remote (Single Source)	Copy	Implementation	Process	Integrity	Accuracy
Slammer Worm	Cracking	Autonomous	Local	Execute	Specification / Design	System	Integrity	Accuracy
Slammer Worm	Cracking	Autonomous	Local	Scan	Specification / Design	Network	Availability	Timeliness
Database Insider: Reconnaissance	Professional Crime /Corporate Rivalry/Vandalism	Human	Local	Probe	Specification	Data	Availability	Timeliness
Database Insider: Data Collection	Professional Crime /Corporate Rivalry/Vandalism	Human	Probe	Read	Specification	Data	Confidentiality& Availability	Precision
BGP route isolation attack	Terrorism /Corporate Rivalry	Human	Remote (Single Source)	Spoof /misdirect /delete /terminate	specification	Network	Availability /integrity	Timeliness /precision

3.2.1.1 UDP Storm

The first cyber-attack is called UDP Storm [10]. In this case, the attacker has learned of two vulnerable machines that will be his victims. The attacker sends out a spoofed packet to the echo port of victim A that appears to come from the echo port of victim B. When victim A receives the packet it responds with an echo-reply to the echo port of victim B. Victim B perceives the packet from A as an echo-request, and sends out an echo-reply to victim A. A then replies to B and this cycle continues until one of the echo services is shut down.

This attack can be executed by anyone who has access to one of many publicly available packet spoofing tools, some knowledge of the fields in a UDP packet, and minimal computer hardware. This is a low enough set of requirements to select Any from the Objective source column of the taxonomy. A single human being starts the attack process, and the point of origin for the packet that triggers the UDP loop comes from a single, remote location. So we select Human from the Propagation category and remote (single source) from the Attack Origin category. This attack is a little unusual in that while only a single packet is sent by the attacker, it creates a packet flood on the network. We select Flood from the action category since that is the attackers intended action. The attacker is taking advantage of a vulnerability in the design of the UDP echo server to create this packet flood, i.e. the designers did not anticipate this kind of abuse and did not include a mechanism to avoid it. Hence we select Specification/Design from the vulnerability category. Through use of the packet flood, the attacker is attempting to increase the ratio of attacking (flood) packets to normal traffic on the network, so we select Network from the Asset category. The attacker changes this ratio by producing a large enough packet flood to reduce the available network bandwidth, so we select Availability from the State effect column. When this ratio gets high enough, the amount of time the network spends carrying legitimate

data is reduced, and the legitimate traffic begins experiencing delays. Thus we select Timeliness from the Performance Effects category.

3.2.1.2 Slammer

The second example is an Internet worm called Slammer [11]. This worm spreads from an infected host by sending out UDP packets to port 1434 at random IP addresses. Each packet contains a buffer overflow attack affecting Microsoft SQL Server 2000, and a complete copy of the worm. When the packet hits a vulnerable machine, a buffer overflow occurs, and this allows the worm to execute on the new victim. Once executing on the new victim, the worm installs itself, and then begins sending out packets to try and locate more hosts to infect.

We have found that worms are best classified by iterating through the classification scheme until all of the details of the attack are expressed. This worm essentially operates in three stages (infect, execute and spread) so we expect to iterate through the classification scheme three times. Based on the skill level involved to implement a worm attack, and the fact that there is no obvious political or financial gain, we consider this to be a cracking Objective with autonomous Propagation. The buffer overflow attack originates at a single remote location relative to the victim machine, and we therefore select Remote (Single Source) from the Attack Origin menu. The first action taken by the worm is to copy itself from the attacker to the victim machine, and this is done by exploiting the implementation (buffer overflow) vulnerability in MS-SQL Server Process. From that information we select Copy and Implementation from the Action and Vulnerability categories respectively, and process in the Asset column. This breaches the integrity of the SQL Server process, and affects the accuracy of its output. That is, the SQL Server process is now under the control of Slammer and will output what the worm tells it to. We therefore select integrity from the State effects and Accuracy from the Performance Effects categories.

We now iterate for the next part of the worm, which takes place on the local machine (victim). Since the attack is now coming from the local system, we select local from the Attack Origin category. At this point, Slammer executes, taking over a process on the victim and we select Execute from the Action category. This ability to execute takes advantage of the fact that the worm is running in the context of the SQL Server process, and the action is allowed by the system specifications. Thus we select Specification/Design from the vulnerability category. The worm is employing the resources of the system it is executing on, and so we select system from the Asset category. When Slammer accesses the system resources, this is a violation of the system integrity, as the worm should not have the privileges to access said resources. This illegal access changes the accuracy of the system because slammer is misidentified as being part of the MS-SQL Server. On this basis, we select Integrity and Accuracy from the State and Performance Effect categories respectively.

In the final iteration, the attack continues to come from the local host, but now Slammer is using the local host to scan for new victims. From these activities, we select Local from the Attack Origin category, and Scan form the Action category. Slammer is operating as the MS-SQL process, and as such, the OS specification continues to allow it to run. So again, we select Specification/Design from the Vulnerability category. From our local point of view on the victim, the worm is now sending out a large amount of network traffic for its scans. This has the effect of reducing available network bandwidth, and hence causing delays in legitimate network

traffic. From this behavior we select Network from the Asset category, along with Availability and Timeliness from the State and Performance Effect categories respectively.

3.2.1.3 Database Insider

For this type of attack, we consider a sample database insider attack. The key aspects of this attack are connection by database user, reconnaissance queries by user, and data collection queries by user. Because, in the database insider problem, much of the relevant information is carried by subtle characteristics of the semantically complex query time series, there are many variants of these aspects. For the insider case, we select Professional Crime, Corporate Rivalry and Vandalism as Objectives. These attacks are propagated by humans from a local location. The Actions are probe and read in each phase respectively. Both phases exploit a Specification/Design vulnerability on a data Asset. Both phases cause an availability State Effect due to user actions on the system. The second phase also breaches confidentiality. For performance effects, we select Timeliness in phase one and Precision in phase two.

3.2.1.4 BGP Route Isolation

A network isolation attack ultimately manifests as complete loss of connectivity to one or more prefixes. The goal of the adversary is to prevent any communication to some victim network. Often, the real target of such an attack is some enterprise or organization within some larger autonomous system. However, because these attacks are carried out against the AS at the prefix level, there is often collateral damage to nearby (in the addressing sense) networks within the same AS. For the purpose of the following analysis, we consider the entire AS to be the victim AS, and defer issues of intra-domain network isolation to future work. The adversary must have access to the control plane of interdomain routing to be able to mount many of the attacks discussed in this section (e.g., is able to force an AS to act maliciously). Given our observation of the relatively poor security practices of many ASes, this does not seem to be unreasonable. We refer to the AS acting maliciously as the adversary AS.

The adversary has several ways to affect the reachability of an AS. Firstly, the adversary can hijack the prefix by claiming to be the origin of the victim's prefixes. Note that because both the victim and adversary AS both continue to announce the prefix, the effect of the attack will only affect those ASes that find the path to the adversary AS to be better (e.g., has shorter path). Because malicious announcements are only propagated to those parts of the network to which they will be the "best origin", sensors should be distributed as many points in the network as is possible.

The adversary can also manipulate the paths through the network to isolate the victim. In the simplest case, the adversary can route all traffic through its AS. The adversary AS could further drop all packets to the network. Other manipulations of BGP can achieve the same effect. The adversary can prevent convergence of the path selection by inserting and removing seemingly good paths, or simply by preventing the propagation of real paths.

Lastly, the adversary can disconnect the AS from the network by physically severing the links between the AS and the larger Internet. Known historically as the backhoe attack, these attacks obviously are beyond the ability of any sensor network to thwart. However, sensors may

be able detect the existence of link severing. Interestingly, link cutting results in many of the same interdomain routing behaviors as the origin and path manipulations described above.

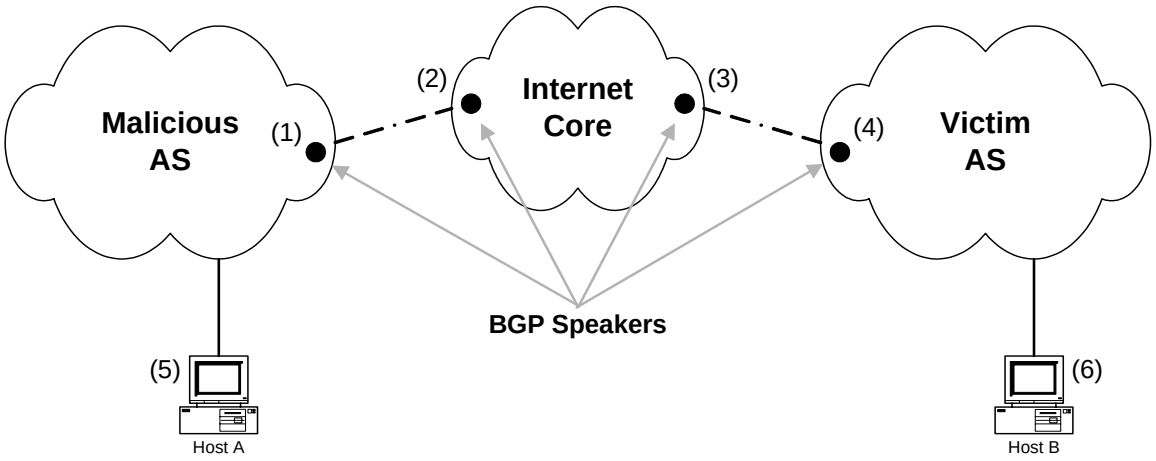


Figure 5. A simplified interdomain routing environment.

Figure 5 presents a view of a simplified network environment. We present only as much of the routing infrastructure to motivate the isolation sensor infrastructure environment. For example, the adversary and victim ASes are presented as stub-ASes, but this is not necessarily the case (e.g., they may transit traffic). Also, there may be any number of other ASes participating in the creation and distribution of routing data. We simply denote this set of ASes as the Internet core. Note also that BGP speakers 2 and 3 and hosts 5 and 6 represent a multiplicity of hosts and nodes upon which we can place sensors.

To classify this attack, we select the following for each column in Table 3: Objective: Terrorism/Corporate Rivalry, Propagation: Human, Attack Origin: remote (single source), Action: spoof/misdirect/delete/terminate, Vulnerability: specification/design, Asset: network, State Effect: availability/integrity and Performance Effect: timeliness/precision.

3.2.2 Classification Trees

Cyber-attack classifications can be depicted in a tabular format as shown above in Table 3, or expanded into a tree structure. The tabular system allows this information to easily be stored in a database, but the tree structure allows quick visual comparisons, and depicts the structure of the cyber attack sub-domains. We employed the tree structure to select representative cyber attacks from each of the sub domains revealed by the categorization scheme. A portion of our tree can be seen as an example in Figure 6.

Threat → Agency → Origin → Action → Vuln. → Target → State → Perf. → Attack

- Any
  - Autonomous Virus / Worm
    - Local
      - Copy
        - Specification / Design
          - Network
            - Availability
              - Timeliness
                - Bugbear.B@mm (Part G - mail/infect other victims) [18]
                - Love Letter worm (Part E - mail/infect other victims) [14]
                - SoBig (Part E - mail/infect other victims) [15]
                - W32.HLLW.Fizzer@mm (Part I - mail/infect other victims) [16]
                - W32.Mimail.A@mm (Part F - mail/infect other victims) [17]
                - W32.Welchia.Worm (Part G - infect other victims) [19]
          - System
            - Confidentiality
              - Precision
                - W32.Mimail.A@mm (Part E - relay information about victim) [17]
            - Integrity
              - Accuracy
                - SoBig (Part F - Download and execute adjunct files) [15]
                - W32.Welchia.Worm (Part C - execute, install on, and Patch victim) [19]
      - Delete
        - Specification / Design
          - System
            - Integrity
              - Accuracy
                - W32.Welchia.Worm (Part E - Remove W32/Blaster) [19]
      - Eavesdrop
        - Specification / Design
          - System
            - Confidentiality
              - Precision
                - Bugbear.B@mm (Part F - relay key log activity from victim) [18]
      - Execute
        - Specification / Design
          - System
            - Integrity
              - Accuracy
                - Bugbear.B@mm (Part C - execute and install on victim) [18]
                - Love Letter worm (Part C - execute on victim, disseminate within file system.) [14]
                - Slammer / MS-SQL Server Worm (Part B - execute on victim) [12]
                - SoBig (Part C - execute and install on victim) [15]
                - W32.HLLW.Fizzer@mm (Part C - execute and install on victim) [16]
                - W32.HLLW.Fizzer@mm (Part E - open back door on victim) [16]
                - W32.HLLW.Fizzer@mm (Part G - disseminate on victim) [16]
                - W32.Mimail.A@mm (Part C - execute and install on victim) [17]
      - Scan
        - Specification / Design
          - Network
            - Availability
              - Timeliness
                - Slammer / MS-SQL Server Worm (Part C - scan/infect other victims) [12]
    - Remote (Single Source)
      - Copy
        - Implementation
          - Process
            - Integrity
              - Accuracy
                - Slammer / MS-SQL Server Worm (Part A - find and infect victim) [12]
  - Human
    - Remote (Single Source)
      - Flood
        - Specification / Design
          - Network
            - Availability
              - Timeliness
                - UDP Storm [1]

Figure 6. Example Classification Tree

What follows is a brief description of the attacks in the example tree. It is important to note that we view the sequence of events and effects from the victim machine. As a result worms often manifest as an incoming attack, and then after establishing themselves on the victim, an outgoing attack.

3.2.2.1 Love Letter

The Love letter worm is primarily an email worm, but it spread by other vectors as well. It is a VBS script, so it can easily be picked up from any infected location. We have broken this cyber attack into five functional stages. In the first stage, a remote host either locates the victim from a stored email address on the infected host, or infects a file that is of interest to the victim. Love Letter then copies itself to the victim machine either through an infected email sent to a user, or via a file downloaded by a user. In the second stage a user on the victim machine is mislead into opening and starting the attachment or file containing the Love Letter worm. The third stage occurs when the worm begins executing, and installs itself on the victim machine. The worm then begins disseminating on the new host, replacing various script files it finds with copies of itself, possibly adding a script to mIRC clients, and updating the start page for explorer. In the fourth stage, the worm reads the address book from the victim machine. In the last stage, the Love Letter emails copies of itself to all of the addresses found in stage four and this may have consequences for network bandwidth [12].

3.2.2.2 SoBig

The SoBig worm spreads strictly by email. We have broken this cyber attack into six functional stages. In the first stage, SoBig is operating on a remote system, has located the victim from an email address read from that remote machine, and sends out an infected email that is received by a user on the victim machine. In the second stage, a user on the system is misdirected into opening the attachment containing SoBig. Then, in the third stage, the worm executes and installs itself on the victim machine. In the fourth stage, the worm scans the victim machine for email addresses. In stage five, SoBig begins propagating by sending email to the addresses found in stage four and this may have consequences for network bandwidth. SoBig can switch in and out of its last stage where it downloads and executes arbitrary files. This has been used to steal confidential information, set up spam relays, and may be used as a method of updating the virus [13].

3.2.2.3 W32.HLLW.Fizzer@mm

Fizzer is primarily an email worm although it will try to spread through the KaZa network by infecting files in the KaZa shared folder. We have broken this cyber attack into nine functional stages. In stage one, the worm locates a victim either from an email address stored on the infected host, or infects a file that is of interest to the victim. Fizzer then copies itself to the victim ether by sending an email to a user, or via a file downloaded by a user on the victim machine. In stage two, a user on the victim machine is misdirected into opening the infected file or attachment. In stage three, Fizzer executes and installs itself on the system. In the next stage, Fizzer will attempt to terminate any anti-virus programs that are on the system. In stage five, the

worm begins the process of opening back doors to the system. This includes connecting to chat rooms (IRC and AOL) to listen for instructions, running as an http server, opening additional ports for connections, and looking for updates. In stage six, Fizzer begins logging keystrokes. In the next stage Fizzer disseminates on the victim, infecting any KaZa shared files that it can. In stage eight, Fizzer scans the victim's machine for email addresses. Finally, in the last stage, the worm begins propagating by email, using the addresses discovered in stage eight and this may have consequences for network bandwidth [14].

3.2.2.4 W32.Mimail.A@mm

The Mimail worm spreads strictly by email and we have broken this cyber attack into six functional stages. In stage one the worm is has located a victim from an email address stored on the infected host, and sent a copy of itself to the victim in an infected email attachment. In stage two, a user on the victim machine is misdirected into opening the infected email attachment. In stage three, Mimail executes and installs on the victim. In stage four the worm scans the victim machine for email addresses that it can use for dissemination. In stage five, Mimail begins capturing data from certain desktop windows on the victim, and relaying that data to its designer via email. In stage six, the worm begins emailing itself to other potential victims (the addresses found in stage four) [15].

3.2.2.5 Bugbear.B@mm

Bugbear is primarily an email worm but also has the capability of spreading over network shares. Our analysis breaks this worm into seven functional stages. To begin, the worm is running on a remote host and either emails itself to a user address found in list on that infected host, or the user on the local (victim) machine has unknowingly downloaded a file infected by the worm. In the next stage, the user is misdirected into opening the infected email or file. In the third stage, Bugbear executes and installs itself on the victim machine. Once stage three is complete, Bugbear spans several worms to execute each of the following stages independently. In stage four, the worm disseminates on the victim, infecting certain files located on the local victim and connected network shares. In stage five, Bugbear opens a back door to the system, and begins listening for commands. In stage six, the Bugbear starts a key logger and periodically sends the log to its creator. In the final stage, the worm scans the victim machine for email addresses, and sends out copies of itself via email to the addresses it finds [16].

3.2.2.6 W32.Welchia.Worm

Welchia is an Internet worm that is fully automated and exploits buffer overflows to spread. For purposes of classification, we have identified seven functional stages in this worm. During stage one of this worms cycle, it uses a ping scan to locate potential hosts. In stage two, Welchia locates a potential host, and exploits a buffer overflow implementation vulnerability in DCOM RPC (or WebDav), to download a copy of itself and the tFTP program from the attacking system. In the third stage, the worm executes on the new victim, where it downloads and installs the RPC patch from Microsoft. In the fourth stage, Welchia reboots the victim machine as part of the patch installation procedure. In stage five, Welchia will halt the blaster

worms process if it is running, and delete blaster.exe thus removing any blaster infection. In stage six, blaster will begin outbound ping scans for new hosts. In stage seven, blaster has located a potential remote host and tries to infect the new machine [17].

3.2.2.7 Slammer / MS-SQL Server Worm

Slammer is described in the classification table example from the previous section.

3.3. Attack Profiling

From our attack classification research, we chose 8 attacks that fit a variety of different fields in our classification table. We created profiles for these attacks.

An attack profile takes the form of a cause-effect chain or network of activities, state changes, and performance impacts with resource-process interactions and activity-state-performance interactions throughout the course of an attack scenario. Attack profiling will help understand attacks and how they can be predicted using observable points, each of which is made up of a data, feature and characteristic (DFC). Each node in the cause-effect network of each attack profile will become a candidate observable point.

Based on the activity-state-performance changes, observable points can be selected such that monitoring the observable points will be useful in predicting the attack. The observable points can be an activity, a state change or a performance impact anywhere along the cause-effect chain for the attack. For each attack, based on the knowledge of the observable points, computer data that can be used to identify/predict the attack, its feature and characteristic can be suggested.

The following definitions and explanations are provided to explain how we profile an attack. After this, we profile one attack with further explanation.

Data: The raw data collected and monitored.

Feature: The measure from the data, such as individual observation, mean, variance, probability distribution, covariance, auto-correlation, dependency, stationarity and chi-square distance.

Characteristic: The characteristic of a given feature that enables the distinction of an attack from normal behavior, such as shift, trend (i.e., cyclic and seasonal), drift (i.e., upward and downward), intermittent spike or bump, and change (step change, spike, exponential change, slope change, sine wave, square wave, etc.), and changes in dynamic state, phase synchronization, etc.

Sub-Indicator: Unique combination of DFC

Indicator: Composed of one or more sub-indicators

Observation: One or more indicators to uniquely identify an attack.

- If indicators have more than one sub-indicator, then **all** sub-indicators are needed to identify the indicator
- If observations have more than one indicator, then **each** indicator independently identifies the observation.
- Indicators will have unique ID numbers. We are developing a novel numbering scheme to uniquely classify indicators. The unique ID for each indicator of an

observation will be based on the indicator's DFC, and where the data is located (including protocol/log and physical location of collection).

Table 4 illustrates the possible variations of indicators and sub-indicators an observation may contain.

Table 4: Illustration of Profile

Observation	Indicator	Data	Feature	Characteristic
A	1			
B	2			
C	3			
	4			
D	5			
	6			

From Table 4, we note the following:

- **Observation A** has only one indicator 1.
- **Observation B** has one indicator 2, which is made of two sub-indicators.
- **Observation C** has two indicators 3 and 4.
- **Observation D** has two indicators 5 and 6. Indicator 6 has two sub-indicators.

Data dependence, Spatial, temporal and causal relationships are captured, in the form of formulas. An attack formula contains the spatial, temporal and causal relationships required to detect a specific attack.

Data dependence relationship: If the same data set is used to identify more than one observation, then these observations have the data dependence relationship. For ex: If C, D and E are all observations made on the same data source, they have the data dependence relationship.

Spatial relationship: Based on locations, such as host (L1), router of hosts network (L2), intermediary network's router (L3), BGP routers AS1, AS2 and AS3.

Temporal relationship: Relationship based on time of occurrence of observations. For ex: A happens before B

Causal relationship: If one observation is the cause for another, a causal relationship exists. For ex: A is the cause for B and C.

Formula to represent the attack: This formula combines the spatial, temporal and causal relationships of an attack. A particular attack formula can be used to specify how to design a sensor to detect that attack. To detect an observation, which has two/more indicators, we could use any/all of the indicators. This is represented by an OR (|) in the formula. Spatial and temporal relationships are captured in the location (l) and time (t) values in the formula. Arrows in the formula indicate a causal relationship.

Next, we describe the development of an attack profile through and example and then profile several attacks as examples. Each attack will have 3 pieces of information: Diagram of observations, attack profile, and formulas: Data Dependency and Spatial/Temporal/Causal.

To describe how this information is displayed, we use the Apache2 Web Server attack as an example. The Apache2 attack is a denial of service process attack. It attacks the web server process by flooding it. In this attack an attacker sends a request with many http headers. If the server receives many of these requests it will slow down, and may eventually crash. The attack is most effective when all the headers are the same. This causes a non-linear consumption of memory, and can quickly overwhelm the available resources [18].

1. Outline of observations (see diagram in following section)

A	HTTP packets with large headers
B	Multiple HTTP packets requesting same file.
C	HTTP requests from one source arrive unusually fast.
D	Comparatively high memory util by web server process
E	Comparatively high CPU util by web server process
F	More HTTP requests arrive than are serviced
G	Web server response time increases

2. Attack profile

The profile for the Apache attack is shown in Table 5, where L_1 = observed at victim machine and L_2 = observed at victim's router. In later profiles, L_3 = observed on the network.

Table 5. Apache Web Server Attack

OBS	Indicator	Data	Feature	Characteristic
A	I_1	EWMA of HTTP GET message header size	Chi-squared distance	Step change
	I_2	EWMA of HTTP GET message header size with same DEST IP	Chi-squared distance	Step change
B	I_1	EWMA of similarity score of pairwise observations of filename in HTTP GET messages	Chi-squared distance	Step change
	I_2	EWMA of similarity score of pairwise observations of filename in HTTP GET messages with same DEST IP	Chi-squared distance	Step change
C	I_1	EWMA of similarity scores of InterArrival Time of HTTP GET messages from same SRC IP	Chi-squared distance	Step change
	I_2	EWMA of InterArrival Time of HTTP GET messages from same SRC IP with same DEST IP	Chi-squared distance	Step change
D	I_1	EWMA of ratio of (Web server memory usage/Sum of all other processes memory usage)	Chi-squared distance	Steady increase
E	I_1	EWMA of ratio of (Web server CPU usage/Sum of all other processes CPU usage)	Chi-squared distance	Steady increase
F	I_1	EWMA of Ratio of count of HTTP GET/POST messages	Chi-squared distance	Steady increase
	I_2	EWMA of Ratio of count of HTTP GET/POST messages to/from same IP	Chi-squared distance	Steady increase
G	I_1	EWMA of difference in arrival times of GET and corresponding POST HTTP messages	Chi-squared distance	Steady increase
	I_2	EWMA of difference in arrival times of GET and corresponding POST HTTP messages to/from same IP	Chi-squared distance	Steady increase

Data collection details

We include data collection details to show how we can collect the required data for an observation. This full table is not included here.

Table 6 only shows an example for reference. The indicator numbers correspond to indicators in the attack profile and will be unique for each indicator (indicators are reusable across attacks).

Table 6. Example data definitions

Indicator	Data	Data Elements
1	EWMA of HTTP GET message header size	HTTP message first header line begins with GET
		HTTP message header size
2	EWMA of HTTP GET message header size with same DEST IP	HTTP message first header line begins with GET
		HTTP message header size
		IP packet header DEST has same IP address
3	EWMA of pairwise observations of filename in HTTP GET messages	HTTP message first header line begins with GET
		HTTP message requested file string
4	EWMA of pairwise observations of filename in HTTP GET messages with same DEST IP	HTTP message first header line begins with GET
		HTTP message requested file string
		IP packet header DEST has same IP address
5	EWMA of InterArrival Time of HTTP GET messages from same SRC IP	HTTP message first header line begins with GET
		HTTP message header field contains "From: <same user>"
6	EWMA of InterArrival Time of HTTP GET messages from same SRC IP with same DEST IP	HTTP message first header line begins with GET
		HTTP message header field contains "From: <same user>"
		IP packet header DEST has same IP address
7	EWMA of difference in arrival times of GET and corresponding POST HTTP messages	HTTP message first header line begins with GET
		HTTP message header begins with POST <same user socket>
8	EWMA of difference in arrival times of GET and corresponding POST HTTP messages to/from same IP	HTTP message first header line begins with GET
		HTTP message header begins with POST <same user socket>
		GET messages to DEST IP
		POST messages from SRC IP = DEST IP of previous row

3. Attack formulas (Data Dependency/Spatial/Temporal/Causal)

***Note: Here we show the entire derivation of the relationship formulas in steps A-E. On the profiles, we will only show A and E.

A. Data dependent relationship:

This relationship is when multiple observations are made on the same data.

Let **X:** A set of attack packets on the network, going to the victim. Then,

$\mathbf{\{x_i , x_j \mid A(x_i), B(x_i , x_j), C(x_i , x_j) \} \forall\ x_i , x_j \in X,}$ where x_i precedes x_j;

- $A(x_i)$ means observation A is made on data element x_i.
- $B(x_i , x_j)$ means observation B is made on the pair (x_i , x_j). In this example, we look at pair wise observations of filename. Thus, we need both x_i and x_j
- Similarly, $C(x_i , x_j)$ is for inter arrival times of two successive packets.

B. Spatial relationships:

$A(l_1) \wedge A(l_2)$, $B(l_1) \wedge B(l_2)$, $C(l_1) \wedge C(l_2)$, $D(l_1) \wedge D(l_2)$, $E(l_1) \wedge E(l_2)$, $F(l_1) \wedge F(l_2)$, $G(l_1) \wedge G(l_2)$

$\wedge$: Observation A is observed at both L1 and L2. Either L1, or L2, or both could be extracted to indicate the observation.

C. Temporal relationships:

$A(t_{i...j})$, $B(t_{i...j})$, $C(t_{i...j})$, $D(t_{i+1...j+1})$, $E(t_{i+1...j+1})$, $F(t_{i+1...j+1})$, $G(t_{i+1...j+1})$, where $0 \leq t_i < t_j$

- Time is represented by t. The subscripts, when compared to one another, give the temporal relationships of the observations.

D. Causal relationship:

The symbol -> is used to indicate: "causes (verb)":

(A, B, C) -> (D, E) -> (F, G)

E. Formula to represent the attack:

This formula is derived from the spatial/temporal/causal formulas. In the case where an observable has two indicators and either one/both can be used to detect the attack, then we say that they have the OR (|) relationship, for purposes of detection. $\wedge$ indicates an AND relationship between observations. So, in the following formula, A and B and C cause D and E which in turn causes F and G:

$$[A(t_{i...j}, l_1) \mid A(t_{i...j}, l_2)] \wedge [B(t_{i...j}, l_1) \mid B(t_{i...j}, l_2)] \wedge [C(t_{i...j}, l_1) \mid C(t_{i...j}, l_2)]$$

$$\downarrow$$

$$[D(t_{i+1...j+1,} l_1) \mid D(t_{i+1...j+1,} l_2)] \wedge [E(t_{i+1...j+1,} l_1) \mid E(t_{i+1...j+1,} l_2)],$$

$$\downarrow$$

$$[F(t_{i+1...j+1,} l_1) \mid F(t_{i+1...j+1,} l_2)] \wedge [G(t_{i+1...j+1,} l_1) \mid G(t_{i+1...j+1,} l_2)]$$

3.3.1 Graphical Representation of Observations

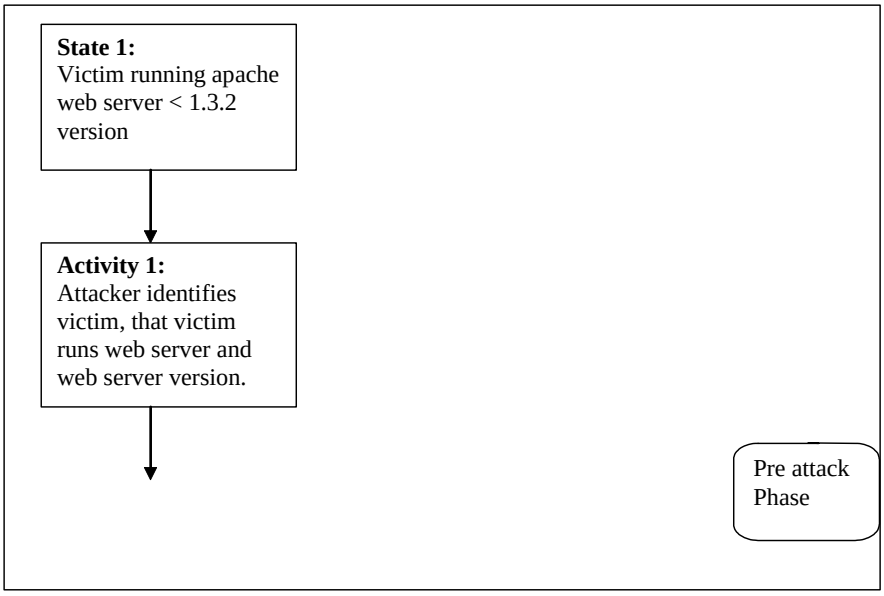
Activity-state-performance interactions are explained in the form of a diagrammatic cause-effect chain. Pre attack phases are indicated for each attack, since predicting an attack is most useful in the pre attack phase itself. Based on the activity-state-performance interactions, possible observable points are also indicated.

3.3.1.1 Apache2 Attack

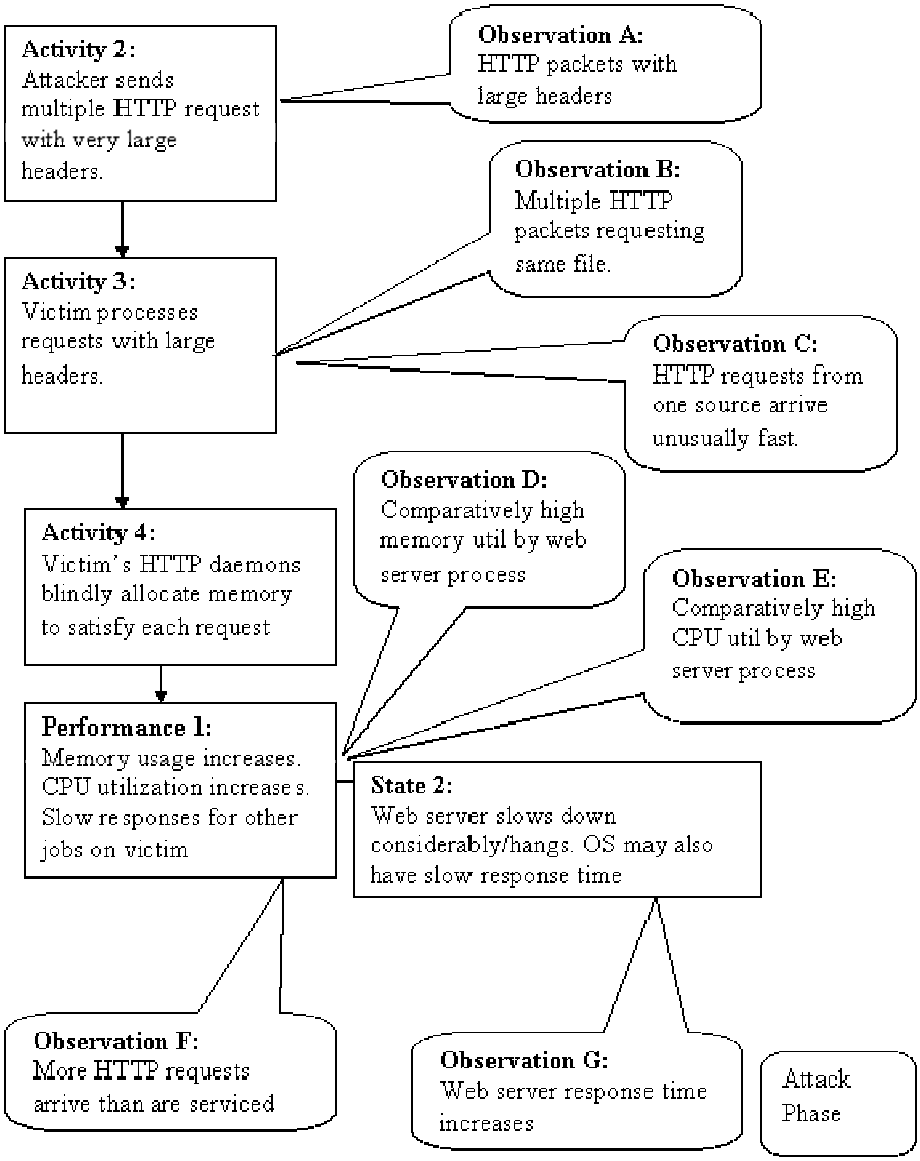
This attack is described in our example above. Here we include the graphic of the cause-effect chain for this attack. Figure 7a describes the pre-attack phase which includes a port scan. We profile this step separately in the NMAP scanner below. Figure 7b is the attack phase of the Apache2 cause-effect profile graphic.

3.3.1.2 Dictionary attack

A dictionary attack is a remote-to-local system attack. It refers to breaking a cipher, or obtaining a password, by running through a list of likely keys, or a list of words. For example, one can 'break' a password on a computer in an English speaking country by encrypting each of a list of English words and comparing each encryption against the stored encrypted version of users' passwords. Since users often choose inappropriate (i.e. easily guessed or broken) passwords, this has historically succeeded about 4 times out of 10 when a reasonable list is used. In the case of a cipher, if keys are suspected to be words, the same technique can be used to break messages encrypted with it [8]. For the purpose of this attack profile, we assume that words are tried in the alphabetical order and with constant time between attempts. Figure 8 shows the Dictionary attack.



a) Pre attack phase of Apache



b. Attack phase of Apache

Figure 7. Apache2 Web Attack

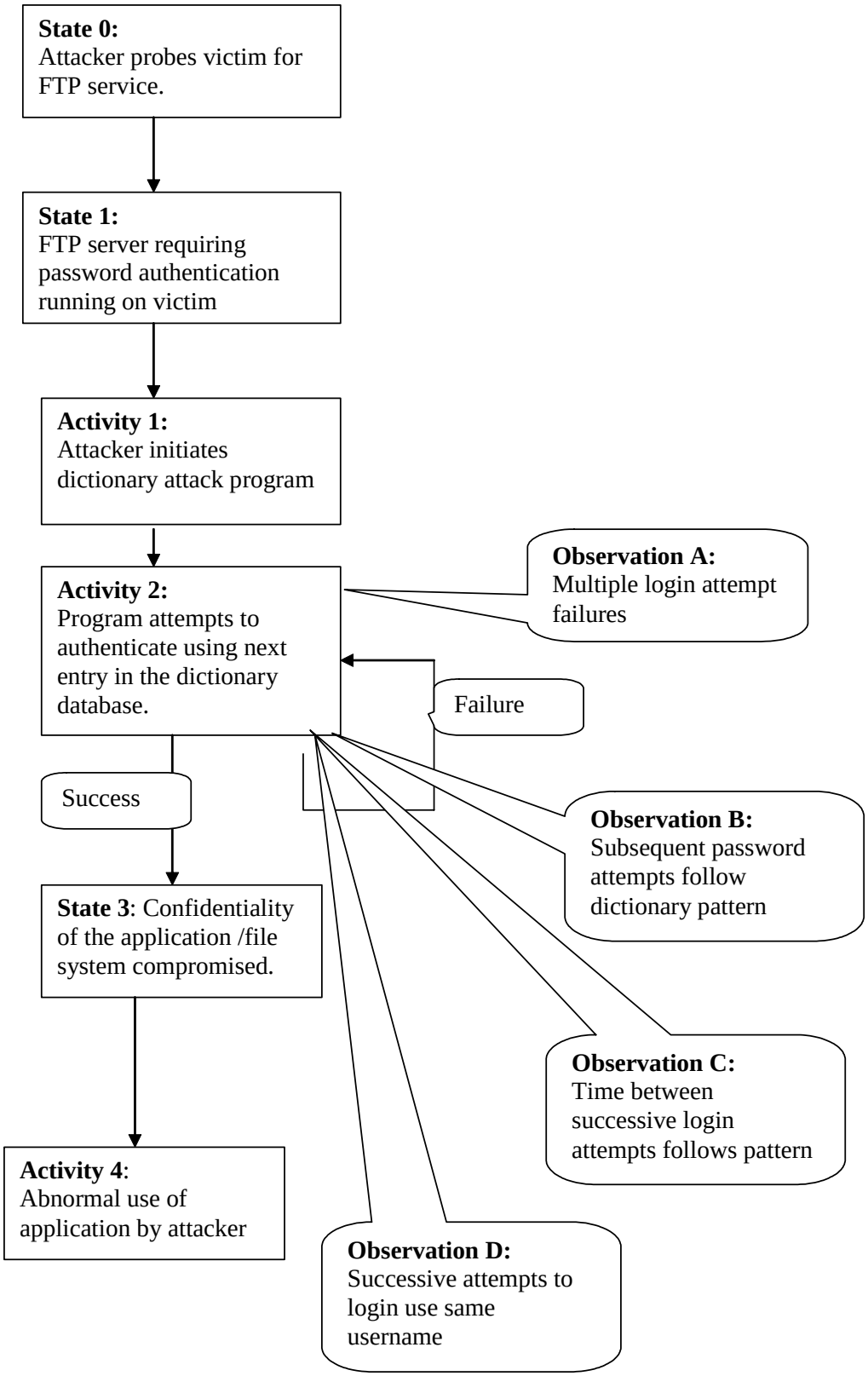
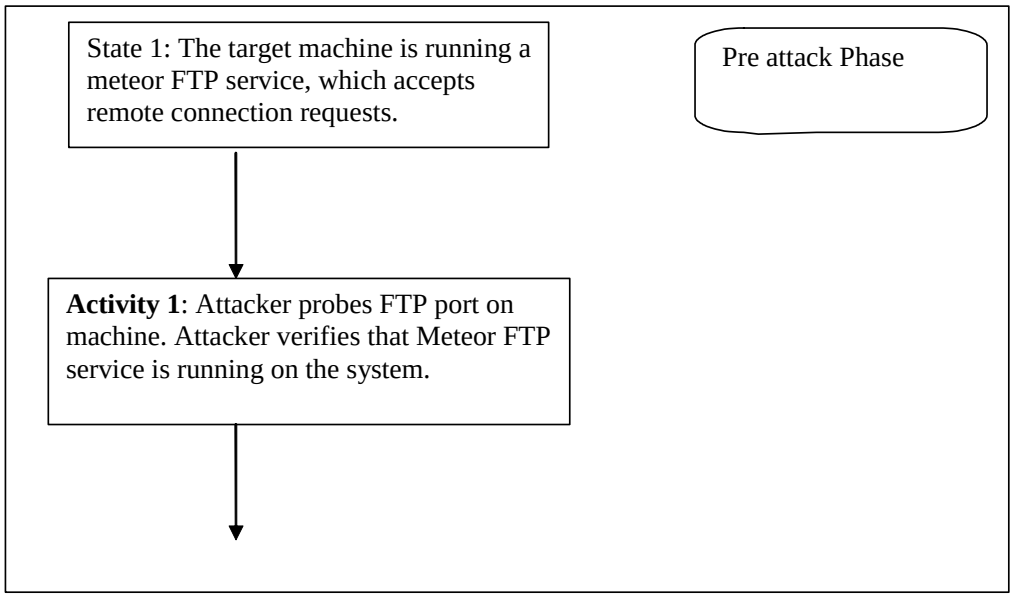


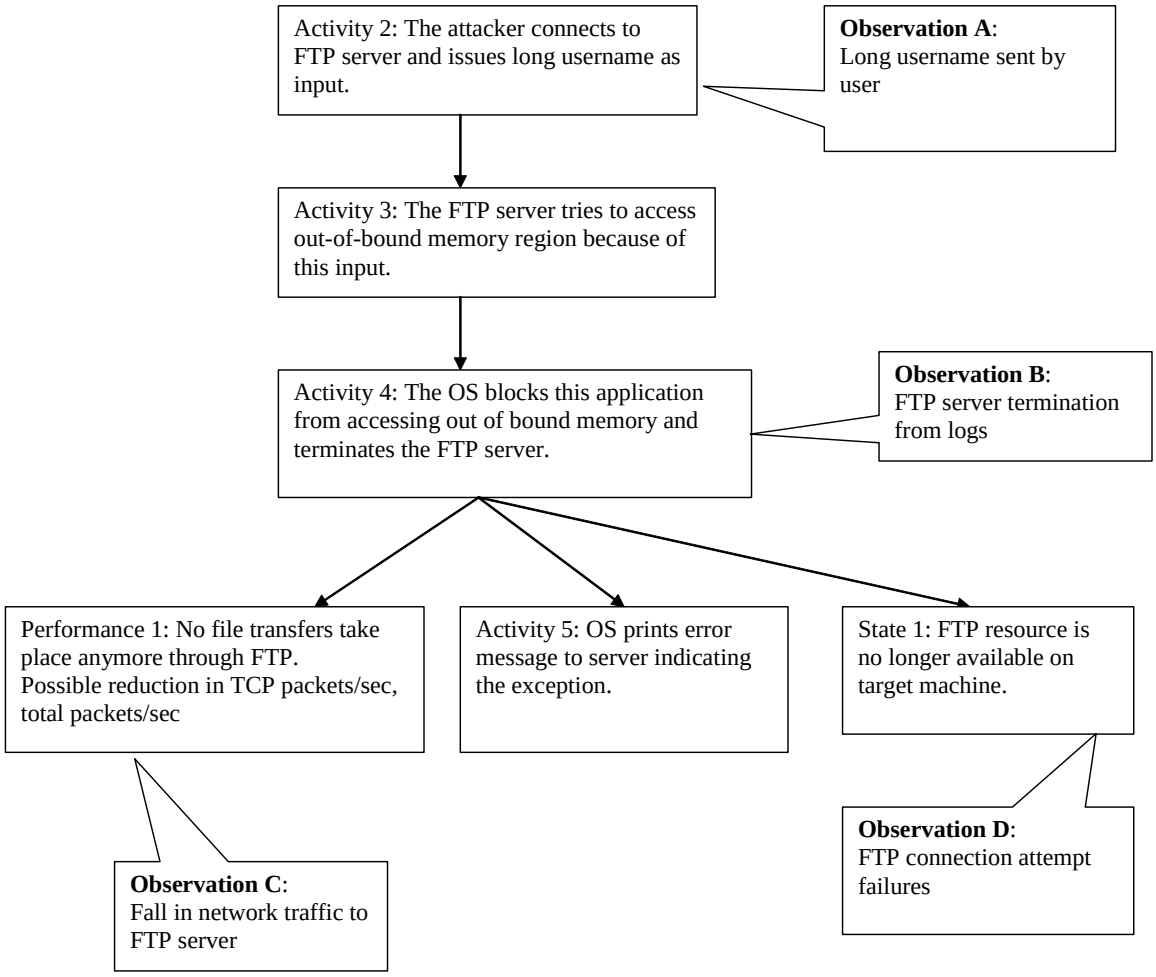
Figure 8. Dictionary Attack

3.3.1.3 Meteor FTP server DoS attack

Meteor FTP server DOS attack is a denial of service process attack. It attacks the ftp server process by exploiting its buffer overflow vulnerability. Meteor FTP server has a buffer overflow vulnerability. If a remote user enters username as USER followed by a random set of characters, the FTP server will crash. This is because the long number of input characters does not get handled properly in the server software [19]. Figure 9 shows this attack.



a) Pre-attack phase of Meteor FTP



b) Attack phase of Meteor FTP

Figure 9. Meteor FTP server DoS Attack

3.3.1.4 NetBus Trojan Attack

NetBus Trojan Attack is a remote-to-local system attack. In this attack, an attacker fools a user into installing a copy of the netbus server on the victim machine. The method used by the attacker to fool the user, is a Trojan containing both the netbus server and the game whack-a-mole. The user is emailed the Trojan, or a link to it, and told that is a free whack-a-mole game. When the user tries to run the whack-a-mole game, an installer starts which installs both the game and the netbus server. The attacker can now use the netbus server as a backdoor to gain access to the system with the same privileges as the user who installed netbus [8]. Figure 10 shows the Netbus attack.

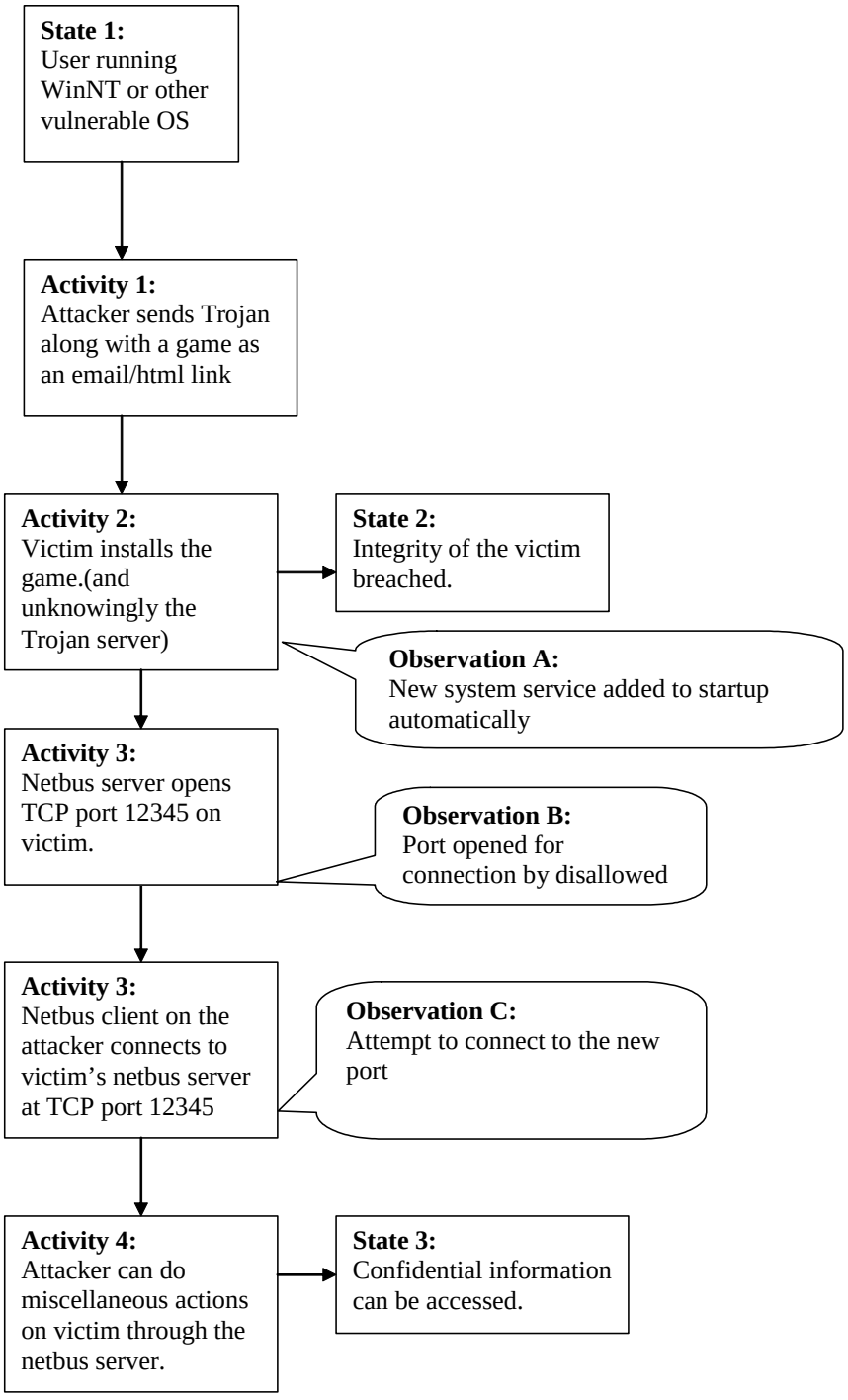


Figure 10. Netbus Trojan Attack

3.3.1.5 NMAP scanner

NMAP Scanner attack is a remote-to-local probe attack. Using a variety of ways, NMAP can identify the OS running, the open ports, the services running on these ports, the version numbers, what types of firewalls are in use and other features. NMAP probes the victim machines with specially crafted IP packets. Based on the responses from the victim machines, it can identify various features of the victim machine/network [8]. Figure 11 shows the NMAP scanner.

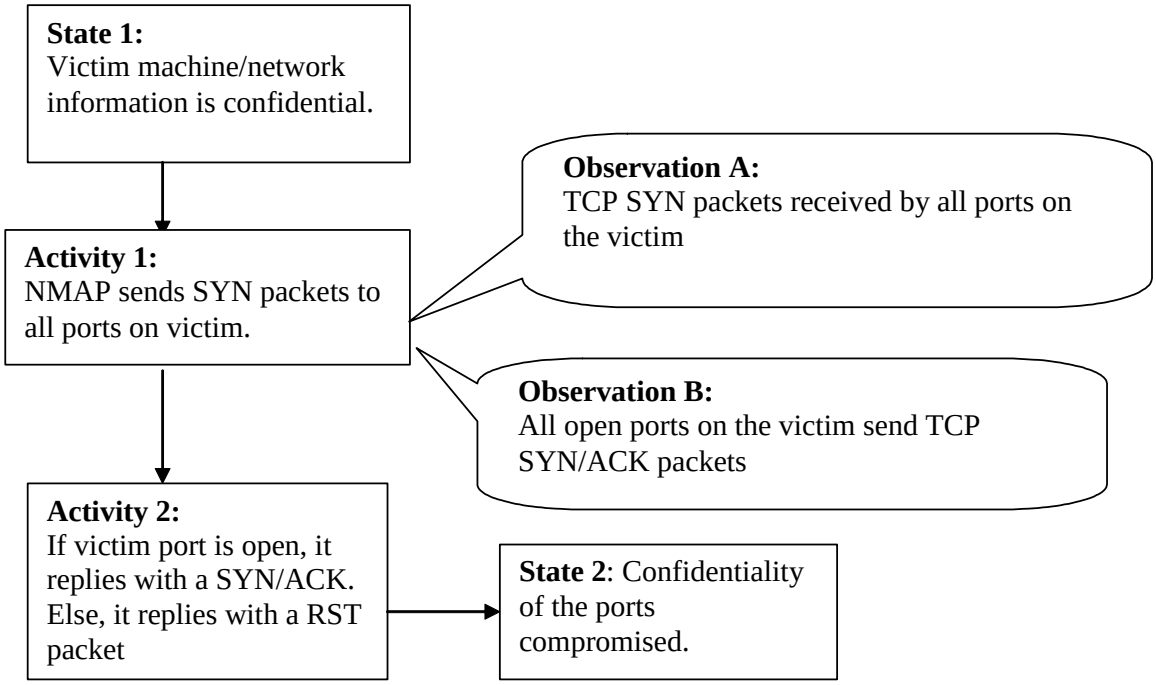
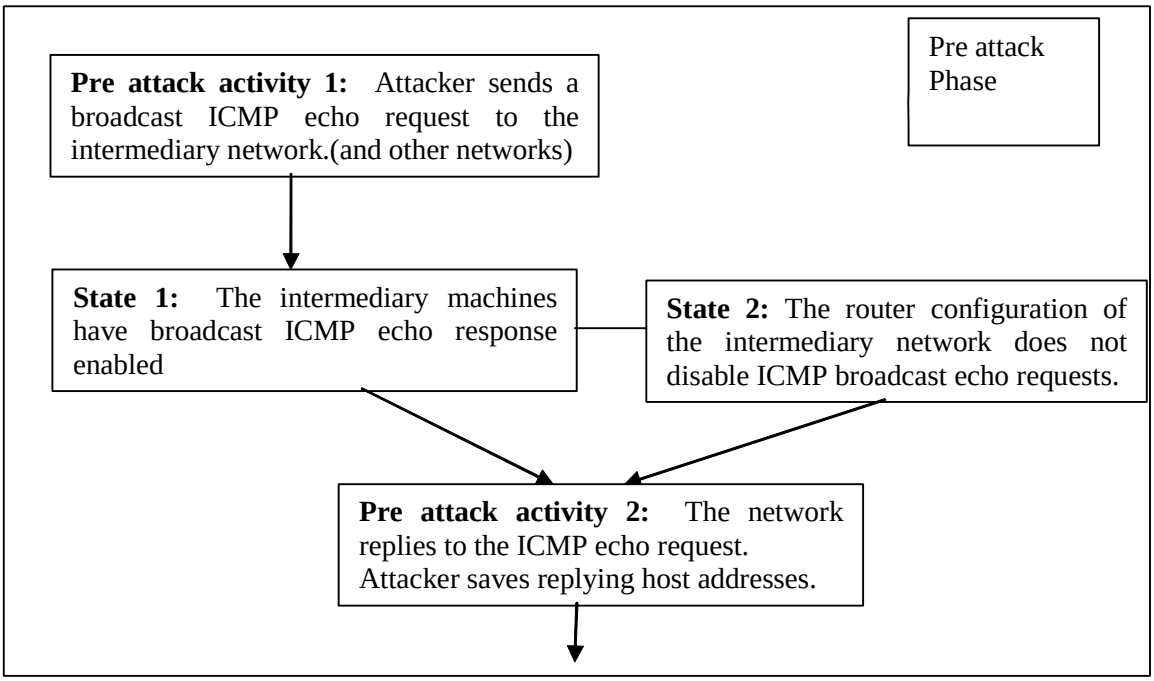


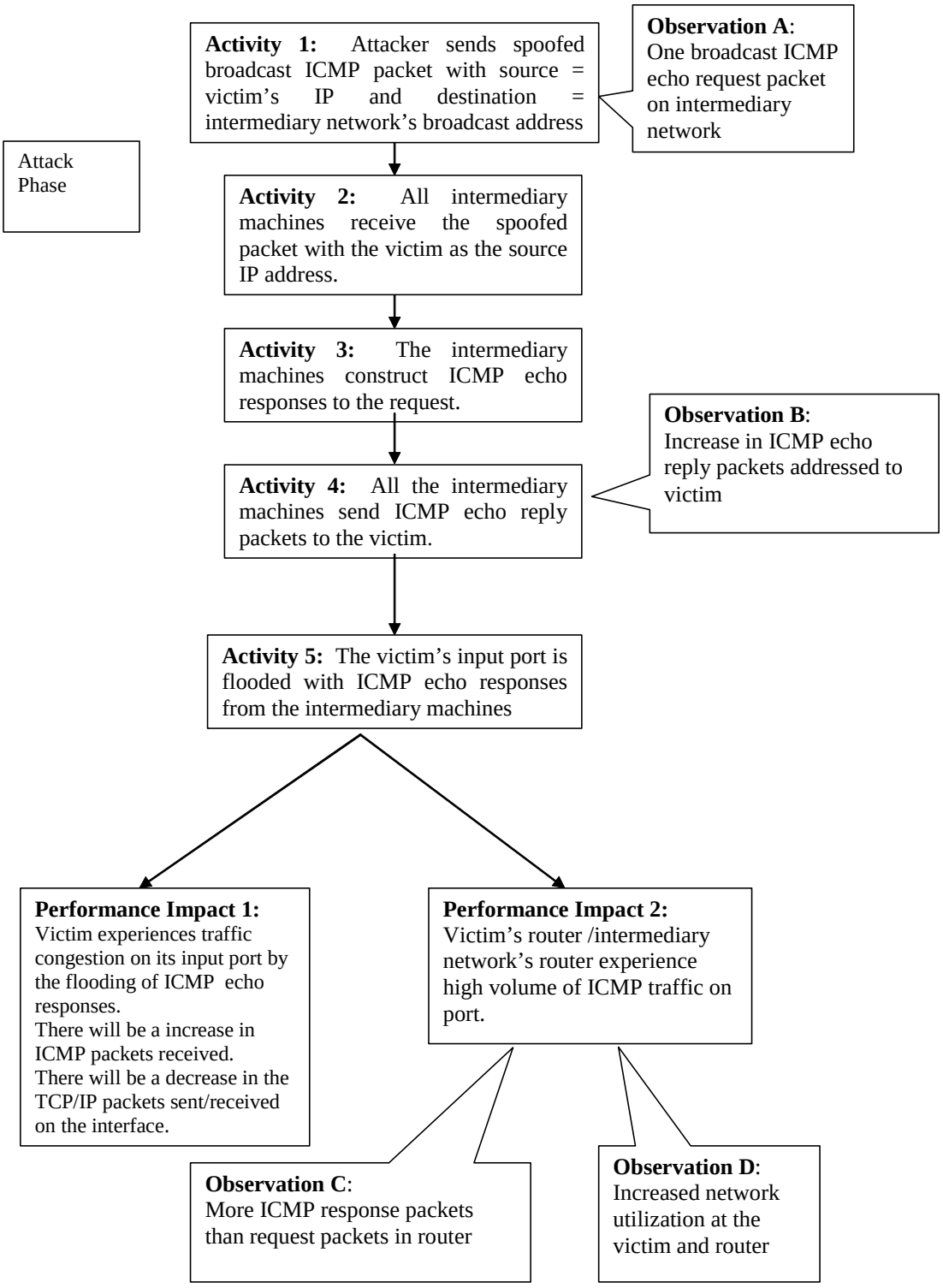
Figure 11. NMAP Scanner

3.3.1.6 Smurf distributed DOS attack

Smurf is a distributed denial of service network attack. In this attack a network connected to the Internet is swamped with replies to ICMP echo (PING) requests. A smurf attacker sends PING requests to an Internet broadcast address. These are special addresses that broadcast all received messages to the hosts connected to the subnet. Each broadcast address can support up to 255 hosts, so a single PING request can be multiplied 255 times. The return address of the request itself is spoofed to be the address of the attacker's victim. All the hosts receiving the PING request reply to this victim's address instead of the real sender's address. A single attacker sending hundreds or thousands of these PING messages per second can fill the victim's network link with ping replies, creating a bottleneck [20]. Figure 12 shows the Smurf attack.



a) Smurf Pre-Attack phase



b) Smurf Attack phase

Figure 12. Smurf Distributed DoS Attack

3.3.1.7 Database Insider

This attack is described in the attack classification section. Figure 13 shows this attack.

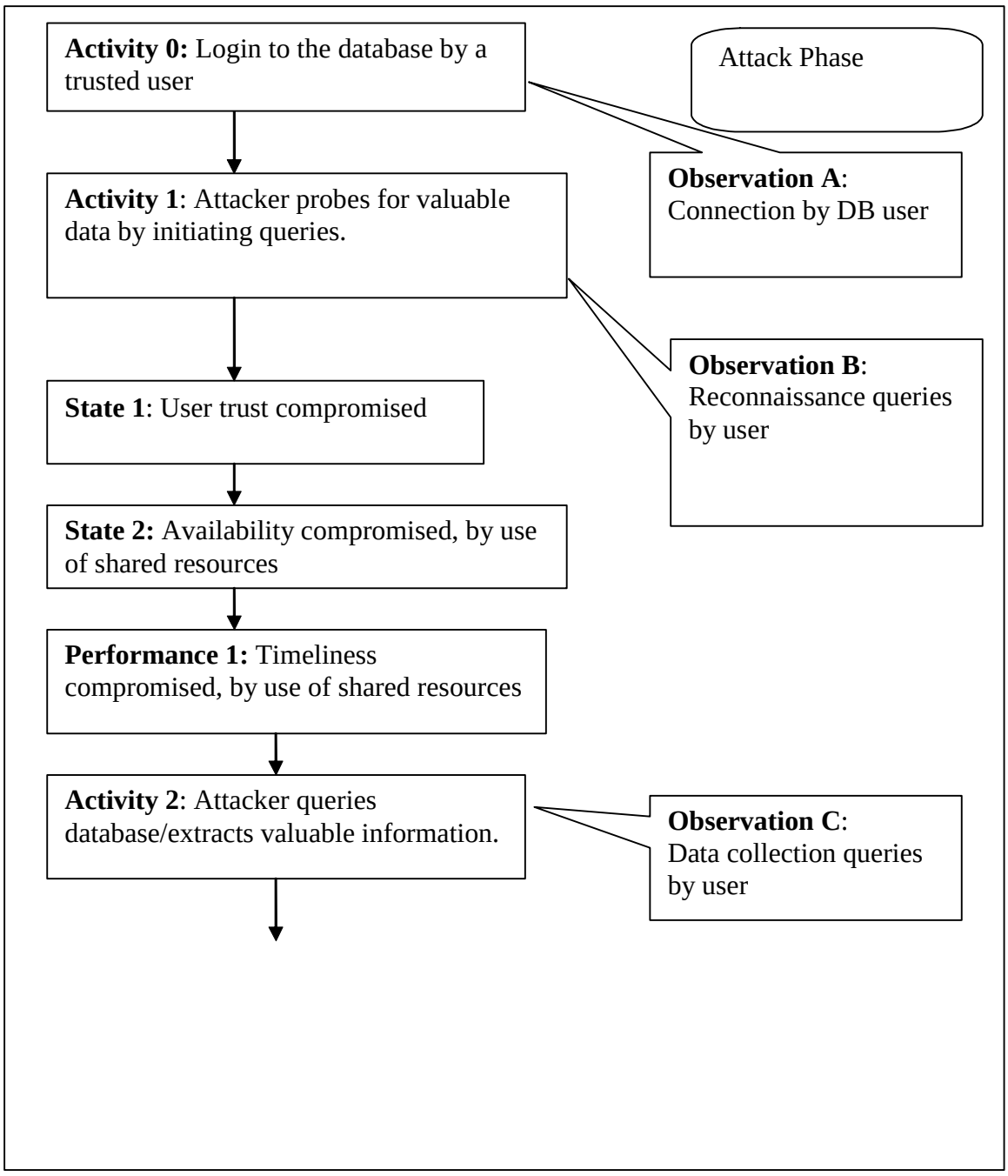
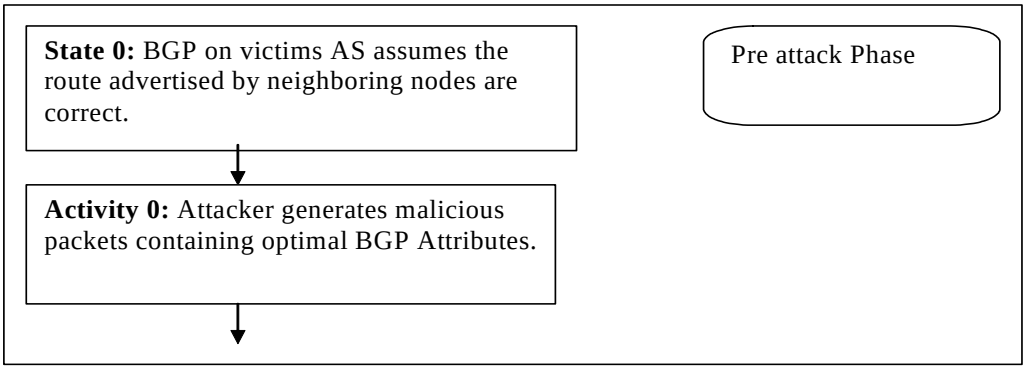


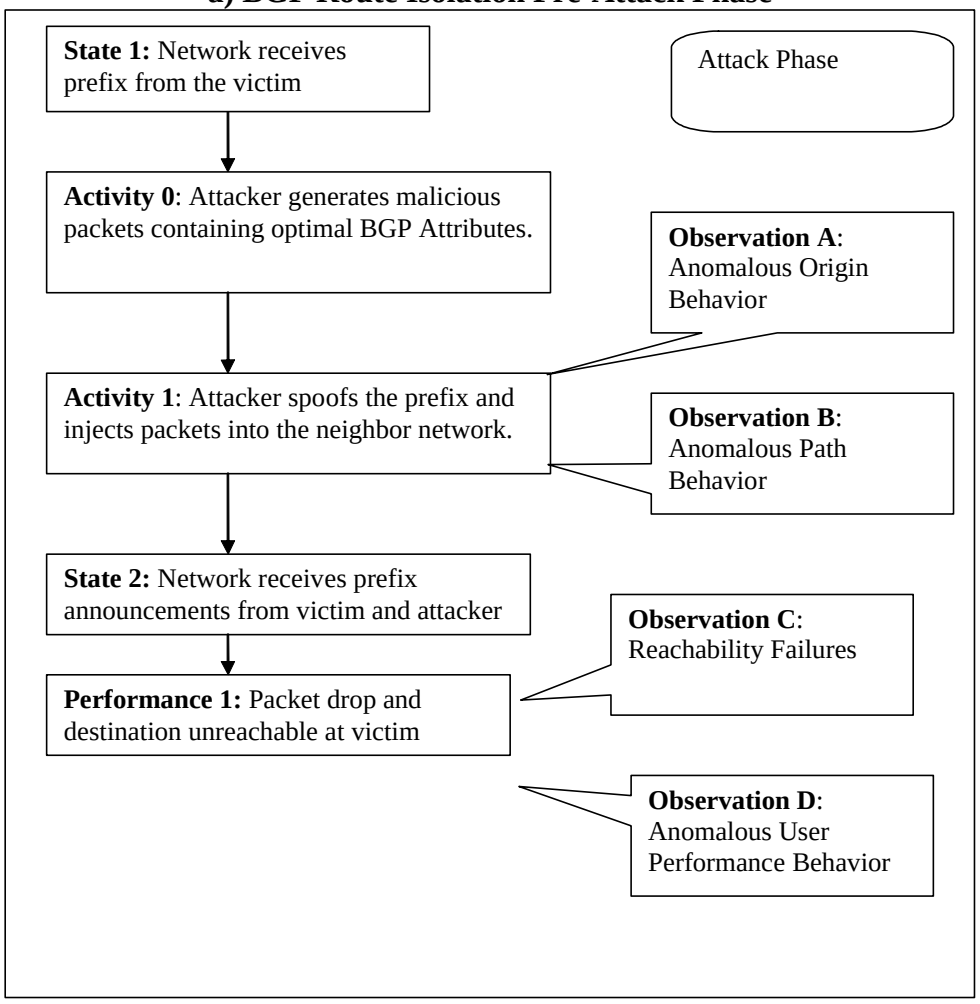
Figure 13. Database Insider Attack

3.3.1.8 BGP Route Isolation

This attack is described in the attack classification section. Figure 14 shows this attack.



a) BGP Route Isolation Pre-Attack Phase



b) BGP Route Isolation Attack Phase

Figure 14. BGP Route Isolation Attack

3.3.2 Profile Table and DFC Relationships

The graphics created in the previous section allow us to view the observations associated with an attack at an abstract layer. For a more comprehensive and detailed look at the attacks, we create an attack profile based on the format given previously. Descriptions of these attacks are as in the previous section. Here we give the attack profiles, along with the relationship formula for our sample attacks. The attacks we are profiling are example attacks. The DFCs we extract are based on current knowledge and previous work. Future research will result in enhancing the features and characteristics of these indicators.

3.3.2.1 Apache2 Attack

This attack is explained in detail above, included as an example for the description of profile tables and DFC relationships.

3.3.2.2 Dictionary attack

Table 7. Dictionary Attack

OBS	Indicator	Data	Feature	Characteristic
A	I_1	EWMA of Number of FTP log entries with keywords "login failure"	Chi-squared distance	Step change
B	I_1	Series of values from *Password* field of login attempt failures in <log>	Sequence pattern	Follows alphabetical order of words in dictionary. Ex: abacus, acme and adjective
C	I_1	Interarrival times of login attempt failures in <log>	Temporal pattern of sequence	Constant time
D	I_1	EWMA of similarity score of pairwise observations of *Username* field of login attempt failures in FTP log	Chi-squared distance	Step change

Observations:

A Multiple login attempt failures
B Subsequent password attempts follow dictionary pattern
C Time between successive login attempts follows pattern
D Successive attempts to login use same username

Data dependent relationship:

Let **X:** A set of FTP log entries on the victim relating to Login failure attempts.

$\{x_i , x_j \mid A(x_i), B(x_{i \ldots i+k}), C(x_i , x_j), D(x_i , x_j) \} \forall x_i , x_j \in X,$

Where x_i precedes x_j, k is a threshold value to be determined.

Attack formula: $[A(t_{i\ldots j} , l_1) \wedge B(t_{i\ldots j} , l_1) \wedge C(t_{i\ldots j} , l_1) \wedge D(t_{i\ldots j,} l_1)]$, where i <j

3.3.2.3 Meteor FTP server DOS attack

Table 8. Meteor FTP DOS Attack

OBS	Indicator	Data	Feature	Characteristic
A	l_1	Length of *username* field in FTP request packet	Individual observation	Greater than a <threshold>
	l_2	Length of *username* field in FTP request packet sent to victim	Individual observation	Greater than a <threshold>
B	l_{1a}	Windows application log	Individual observation	Has value "Access Violation at <memory> Program Terminated"
	l_{1b}	FTP server log	Individual observation	Has value "server process terminated by operating system"
C	l_1	EWMA of TCP packets/sec	Chi-squared distance	Step change
	l_2	EWMA of TCP packets/sec to FTP server's IP address	Chi-squared distance	Step change
D	l_1	EWMA of Count of TCP RST packets from *SRC_PORT* = <ftp port>	Chi-squared distance	Step change
	l_2	EWMA of Count of TCP RST packets from *SRC_PORT* = <ftp port> from victim IP	Chi-squared distance	Step change

Observations:

A Long username sent by user
B FTP server termination from logs
C Fall in network traffic to FTP server
D FTP connection attempt failures

Data dependent relationship

There is no data dependent relationship amongst the observations for this attack

Attack Formula:

$[A\ (t_i, l_1) \mid A(t_i, l_2)] \rightarrow [B\ (t_j, l_1)] \wedge [C(t_{j\ldots k}, l_1) \mid C(t_{j\ldots k}, l_2)] \wedge [D\ (t_{j\ldots k}, l_1) \mid D\ (t_{j\ldots k}, l_2)]$
where $i < j < k$

3.3.2.4 NetBus Trojan Attack

Table 9. Netbus Trojan Attack

OBS	Indicator	Data	Feature	Characteristic
A	l_1	Windows security log	individual observation	has value "New Objects added to registry"
		Windows security log	individual observation	has value "New system service started"
B	l_1	List of open ports on the host from <log>	individual observation	<port> added to the list
		List of open ports on the host from <log>	individual observation	<port> opened by an application not in the allowed list
C	l_1	*Port* field of TCP SYN packet	individual observation	<port>
	l_2	*Port* field of TCP SYN packet sent to victim	individual observation	<port>

Observations:

A New system service added to startup automatically
B Port opened for connection by disallowed application on the host
C Attempt to connect to the new port

Data dependent relationship:

There is no data dependent relationship amongst the observations for this attack

Attack formula: $A(t_i, l_1) \rightarrow B(t_j, l_1) \wedge [C(t_k, l_1) \mid C(t_k, l_2)]$

3.3.2.5 NMAP scanner

Table 10. NMAP Scanner

OBS	Indicator	Data	Feature	Characteristic
A	l_{1a}	EWMA of number of ports receiving TCP SYN packets in time interval t	Chi-squared distance	Step change
	l_{2a}	EWMA of number of ports receiving TCP SYN packets in time interval t on victim	Chi-squared distance	Step change
	l_{1b}	EWMA of number of ports receiving TCP RST packets in time interval t	Chi-squared distance	Step change
	l_{2b}	EWMA of number of ports receiving TCP RST packets in time interval t at victim	Chi-squared distance	Step change
	l_{1c}	EWMA of Ratio of count of TCP SYN to ACK messages received	Chi-squared distance	Step change
	l_{2c}	EWMA of Ratio of count of TCP SYN to ACK messages to victim	Chi-squared distance	Step change
	l_{1d}	EWMA of ratio of number of ports sending SYN/ACK packets to ports sending RST packets	Chi-squared distance	Step change
	l_{2d}	EWMA of ratio of number of ports sending SYN/ACK packets to ports sending RST packets from victims IP	Chi-squared distance	Step change
B	l_{1a}	EWMA of number of ports sending TCP SYN/ACK packets in time interval t	Chi-squared distance	Step change
	l_{2a}	EWMA of number of ports sending TCP SYN/ACK packets in time interval t on victim	Chi-squared distance	Step change
	l_{1b}	EWMA of number of ports sending TCP RST packets in time interval t	Chi-squared distance	Step change
	l_{2b}	EWMA of number of ports sending TCP RST packets in time interval t on victim	Chi-squared distance	Step change
C	l_1	EWMA of number of ports with TCP connections in time interval t	Chi-squared distance	Step change
	l_2	EWMA of number of ports with TCP connections in time interval t at victim	Chi-squared distance	Step change
D	l_{1a}	EWMA of number of ports receiving a string from NMAP list of probe strings in time interval t	Chi-squared distance	Step change
	l_{2a}	EWMA of number of ports receiving a string from NMAP list of probe strings at victim in time interval t	Chi-squared distance	Step change
	l_{1b}	EWMA of number of ports sending a string from NMAP list of match strings in time t	Chi-squared distance	Step change
	l_{2b}	EWMA of number of ports sending a string from NMAP list of match strings at victim in time inteval t	Chi-squared distance	Step change

Observations:

A TCP SYN packets received by all ports on the victim

B All open ports on the victim send TCP SYN/ACK packets

C TCP connection established at all open ports on the victim

D Subset of open ports receive one/more well known NMAP probes

Data dependent relationship:

There is no data dependence relationship among the list of NMAP observations

Attack formula:

$[A (t_{i...j}, l_1) \mid A (t_{i...j}, l_2)] \rightarrow [B(t_{i...j}, l_1) \mid B(t_{i...j}, l_2)], [C (t_{k...l}, l_1) \mid C (t_{k...l}, l_2)],$
$[D (t_{k...l}, l_1) \mid D (t_{k...l}, l_2)]$, where $i < j < k < l$

3.3.2.6 Smurf distributed DOS attack

Table 11. Smurf DoS Attack

OBS	Indicator	Data	Feature	Characteristic
A	I_3	*Type* field of broadcast ICMP packet	Individual observation	Has value 8 (Echo request)
B	I_1	EWMA of interarrival time of ICMP echo reply packets	Chi-squared distance	Step change
	I_2	EWMA of interarrival time of ICMP echo reply packets to the same *DEST*	Chi-squared distance	Step change
C	I_2	EWMA of the ratio of ICMP rsp/sec to req/sec at router	Chi-squared distance	Step change
D	I_1	EWMA of IP packets received/sec from performance log	Chi-squared distance	Step change
	I_2	EWMA of IP packets received/sec from the router's log	Chi-squared distance	Step change

Observations:

A One broadcast ICMP echo request packet on intermediary network
B Increase in ICMP echo reply packets addressed to victim
C More ICMP response packets than request packets in router
D Increased network utilization at the victim and router

Data dependent relationship:

There is no data dependence relationship among the list of NMAP observations

Attack formula: $A (t_i, l_3) \rightarrow [B (t_j, l_1) \mid B (t_j, l_2)] \wedge C (t_j, l_2) \rightarrow [D (t_j, l_1) \mid D (t_j, l_2)]$

3.3.2.7 Database Insider

Table 12. Database Insider Attack

OBS	Indicator	Data	Feature	Characteristic
A	1	Time of connection and *database user ID*	Individual observation	Outside historical range
	2	EWMA of duration of connection/session by user	Chi-squared distance	Step change
B	3	Number of tables outside historic range accessed by user within time interval t	Individual observation	Greater than <threshold>
	4	EWMA of InterArrival Time of queries from the same user	Chi-squared distance	Step change
	5	EWMA of queries selecting no data within time interval t (i.e., requested data not found)	Chi-squared distance	Step change
	6	EWMA of size of query text	Chi-squared distance	Step change
	7	EWMA of ratio of SELECT queries to other queries within time interval t	Chi-squared distance	Step change
	8	EWMA of pairwise semantic distances between relations and attributes of successive queries	Chi-squared distance	Step change
	9	EWMA of Number of queries using stored procedure in time interval t	Chi-squared distance	Step change
	10	EWMA of Number of SQL constructs used in queries in time interval t	Chi-squared distance	Step change
C	11	Number of tables outside historic range accessed by user within time interval t	Individual observation	Below <threshold>
	12	EWMA of Interarrival times of queries to same table	Chi-squared distance	Step change
	13	EWMA of Number of queries using stored procedure in time interval t	Chi-squared distance	Step change
	14	Ratio of rows retrieved during session to total rows in table	Mean	Increase
	15	Ratio of columns retrieved during session to total columns in table	Mean	Increase
	16	EWMA of Interarrival times of queries	Chi-squared distance	Step change
	17	EWMA of pairwise semantic distances between relations and attributes of successive queries	Chi-squared distance	Step change
	18	EWMA of number of SQL constructs	Chi-squared distance	Step change

Observations:

A Connection by database user

B Reconnaissance queries by user

C Data collection queries by user

Data dependent relationship:

There is no data dependent relationship amongst the observations.

Attack formula: $A(t_{i\ldots j,}\, l1) \rightarrow B(t_{k\ldots l,}\, l1) \rightarrow C(t_{m\ldots n,}\, l1)$, where $i<j<k<l<m<n$

3.3.2.8 BGP Route Isolation

In this table we use AS with subscripts to indicate observations made at the AS points identified in the description of this attack above.

Table 13. BGP Route Isolation Attack

OBS	Indicator	Data	Feature	Characteristic
A	L_2, AS_{1-3}	*Prefix* field of BGP withdrawal	Individual observation (withdrawal)	Has unexpected value
	L_2, AS_{1-3}	*Prefix* and *AS number* of BGP update	Individual observation (update)	Has illegal/unknown prefix, AS number
	L_2, AS_{1-3}	Origin change distribution	Prefix change distribution	Non-fitting distribution
B	L_2, AS_1, AS_2	BGP withdrawal	Individual observation (update)	Expected path loss
	L_2, AS_1, AS_2	*Path selection* made in BGP update	Individual observation (update)	Unexpected path selected
	L_2, AS_1, AS_2	Path change	Path change distribution	Non-fitting distribution
	L_2	*Path selection* made in BGP update	Individual observation (update)	Sub-optimal path selected
	L_2, AS_3	BGP update	Individual observation (update)	Illegal AS traffic transiting
C	L_2, AS_1, AS_2, AS_4	Traceroute data log	Individual observation	Has "network unreachable" message
	L2, AS1, AS2,AS4	TCP connection state	Connection results	All new connections dropped (FIN/RST)
D	AS_4, L_1	EWMA of UDP packets/sec from logs	Chi-squared distance	Step change
	AS_4, L_1	EWMA of TCP connections failed/sec from logs	Chi-squared distance	Step change
	AS_4, L_1	EWMA of TCP connections reset/sec from logs	Chi-squared distance	Step change

Observations:

A Anomalous Origin Behavior
B Anomalous Path Behavior
C Reachability Failures
D Anomalous User Performance Behavior

Data dependent relationship:

There is no data dependent relationship among the observations

Attack formula:

$A\ (t_{i\ldots j,}\ l_2) \mid A\ (t_{i\ldots j},\ AS_{1..3}) \rightarrow B\ (t_{k\ldots l},\ l_2) \mid B\ (t_{k\ldots l},\ AS_{1\ldots3}) \rightarrow C(t_{m\ldots n}\ ,\ l_2) \mid C(t_{m\ldots n}\ ,\ AS_{1,2,4}) \wedge D(t_{m\ldots n}\ ,\ l_1) \mid D(t_{m\ldots n}\ ,\ AS_4)$

3.4 Summary

In this section we have outlined our work to catalogue attacks on computer and network systems. We use these tools to design sensor models for cyber attacks. In the next section we use this knowledge of cyber attacks and conduct analysis to discover characteristics of each the observable points of cyber attacks and noise.

4. Characteristics of Cyber Signal and Noise

The cyber signal detection approach engages in the scientific discovery of DFCs for cyber signal (attack data) and noise (normal data). We use attack profiling and data analysis techniques to generalize the DFCs that exist in cyber signal and noise data. We also leverage well-established signal detection models in the physical space (e.g., radar signal detection), and verify them in the cyber space. With this foundation of information, we can build cyber signal detection models that incorporate the characteristics of both cyber signals and noise. This enables us to take the least amount of relevant data necessary to achieve detection accuracy and efficiency. The cyber signal detection approach considers not only activity data, but also state and performance data along cause-effect chains of cyber attacks on computers and networks. We aim to achieve the detection adequacy lacking in existing intrusion detection systems.

To build a cyber signal detection model for detecting cyber attacks on computer and network systems, we need to obtain an understanding of attack and normal data characteristics; otherwise, we cannot have full confidence in detection accuracy, and we cannot achieve data relevancy and have detection efficiency. With knowledge of attack and normal data (cyber signal and noise) characteristics, we can easily leverage signal detection algorithms to build models if discovered cyber signal and noise characteristics also appear in models of the physical domain; or we can easily build new signal detection models using the methodologies of model building in the physical domain if discovered characteristics are not found in the physical domain.

In this section, we provide descriptions, observation points and DFC tables for the six attacks and one worm investigated in this study. We then show an example of generalizations of observations across attacks. Next we provide an overview of signal detection in the physical space including a literature review, summary of our findings, and mapping the physical space to cyber space.

The remaining sections outline experiments and results for discovering cyber signal characteristics. We describe the simulation and data collection of the six attacks (EZPublish Confidentiality, NMAP Scanner, Netbus Trojan, Meteor FTP, IRC Chat Server Abuse and ARP Poison) and one worm (Sobig) in our study. Next we give the results of our analysis to describe cyber attack and noise characteristics and finally, conclude this section.

4.1. DFC for Attacks/Worm in this Section

For this study we experimented with six attacks and one worm. Three of these attacks are described previously in this report: NMAP Scanner, Netbus Trojan and Meteor FTP. In this section, we present the other three attacks and worm: EZPublish, IRC Chat, ARP Poison and Sobig. We give each attack's associated list of observation points and DFC table. For an in depth look at the topics covered in this section, we refer to [21-24].

4.1.1 EZPublish Confidentiality Attack

We describe attacks as a series of observation points. For example, Table 14 shows the observation points for the EZPublish attack.

Table 14. EZPublish Observation Points

Observable point	Observation
A	File from restricted directory accessed
B	File with system related extension accessed

We define each observation point in terms of the data under consideration and its feature and characteristic to distinguish attacks from normal scenarios [21]. Table 15 shows the DFC for the EZPublish attack [24].

Table 15. EZPublish DFC

OBS	Location	Data	Feature	Characteristic
A	L1	*Filename* of HTTP GET request	Individual observation	String match with *settings*
	L2	*Filename* of HTTP GET request to host	Individual observation	String match with *settings*
B	L1	*Filename* of HTTP GET request	Individual observation	String match with *.ini*
	L2	*Filename* of HTTP GET request to host	Individual observation	String match with *.ini*

In the first column of Table 15, we see the letter of the observation that each DFC corresponds to. In the next column, we see the location where the data is collected, L1 is the victim (host) and L2 is a router in the victim's network. The third column describes the actual data we need to collect for this observation point. The fourth column is the feature we need to extract from this data and the last column is the characteristic on the feature, which identifies this observation point.

4.1.2 IRC Chat Server Abuse

Table 16 gives the observation points for the IRC Chat Server Abuse attack.

Table 16. IRC Chat Observation Points

Point	Observation
A	Distrusted applications installed on client and server
B	New service starts on server machine
C	New port opened for service on server
D	Increase in TCP connections at server
E	Ping packets from clients to server at regular intervals

Table 17 gives the DFC for IRC Chat.

Table 17. IRC Chat DFC

Obs	Location	Data	Feature	Characteristic
A	L1	Application log	Individual entry	Has value “IRC chat server installed”
	L2	Application log	Individual entry	Has value “IRC chat client installed”
B	L1	Security log	Individual entry	Has value “ new service started”
C	L1	Open ports on system	List of ports	New port added
D	L1	*TCP connections/sec* at server	Exponentially weighted moving average	Step increase
	L3	*TCP connections/sec* with *DEST* = server IP	Exponentially weighted moving average	Step increase
E	L1	*ICMP requests/sec* at server	Exponentially weighted moving average	Step increase
	L3	*ICMP requests/sec* with *DEST* = server IP	Exponentially weighted moving average	Step increase

4.1.3 ARP Poison

Table 18 gives the observation points for the ARP Poison attack. Table 19gives the DFC for ARP Poison.

Table 18. ARP Poison Observation Points

Observation point	Observation
A	Updates received from different machines do not match.
B	Victim is unable to reach destination
C	Outbound Network traffic reduces

Table 19. ARP Poison DFC

OBS	Location	Data	Feature	Characteristic
A	L1	*Target HA* field of ARP Reply packets received	String Comparison of values	All updates do not have the same value
	L2	*Target HA* field of ARP Reply packets received	String Comparison of values	All updates do not have the same value
B	L1	*Network interface:packet outbound errors* field in performance log	Exponentially weighted moving average	Step increase
	L2	*DEST MAC* field of Ethernet frames - and - Contents of ARP table	String comparison of values	Value in field does not match any entries from ARP table
C	L1	*Network interface: Bytes set/sec* field from performance log	Exponentially weighted moving average	Step decrease

4.1.4 Sobig Worm

Table 20 gives the observation points for the Sobig Worm. Table 21 gives the DFC for Sobig. In The location column, L4 refers to the intermediate mail server involved in the attack.

Table 20. Sobig Observation Points

Observable Point	Observation
A	E-mail infected with Sobig worm received in user's mailbox
B	New process, the Sobig worm, started by user
C	New files created by worm
D	New values added to the startup keys of the registry
E	New worm process starts and original worm process is terminated
F	New event created
G	New threads created
H	Higher CPU utilization as threads search for e-mail addresses
I	Higher file system activity as threads search for e-mail addresses
J	Increased network activity as threads send out infected e-mails
K	Mail activity at mail server as infected e-mails arrives
L	UDP packets sent to update servers

Table 21. Sobig DFC

OBS	Location	Data	Feature	Characteristic
A	L1	E-mail contained in the user's mailbox	Individual observation	9 possible subject lines, 9 possible attachment file names, and 2 possible body lines
B	L1	Name of new process running	Individual observation	9 possible names
C	L1	Filename of newly created file	Individual observation	"%Windir% \winppr32.exe"
		Filename of newly created file	Individual observation	"%Windir% \winstt32.dat"
D	L1	Value of data added to registry key HKEY_LOCAL_MACHINE\SOFTWARE \Microsoft\Windows\CurrentVersion\Run	Individual observation	"TrayX"= "%Windir% \winppr32.exe /sinc"
		Value of data added to registry key HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Windows\CurrentVersion\Run	Individual observation	"TrayX"= "%Windir% \winppr32.exe /sinc"
E	L1	Name of old process terminated	Individual observation	9 possible names
		Name of new process running	Individual observation	"winppr32.exe"
F	L1	Value of event created	Individual observation	"TrayX"
G	L1	9 new threads created	Individual observation	Thread count increment
H	L1	Comparatively high CPU utilization by winppr32.exe	Mean	Increase
I	L1	Comparatively high file system utilization by winppr32.exe	Mean	Increase
J	L1	IP packets sent/sec from performance log	EWMA	Step increase
	L2	IP packets received/sec and sent/sec from the router's log	EWMA	Step increase
	L4	IP packets received/sec from performance log	EWMA	Step increase
K	L4	E-mails received/sec	EWMA	Step increase
L	L1	Destination IP addresses and port used for 20 UDP packets sent	Individual observation	20 IP addresses over port 8998

4.2 DFC Generalization from Profiling

The DFC tables from attack profiling, along with additional attack profiles from [24], give us a starting point for generalizing DFC for cyber attacks. Table 22 shows a sample of some generalizations made on these attack profiles.

Table 22. Generalized DFC from attack profiles [24]

Generalized Description	Attack	Attack Specific Description
The length of the buffer to hold the packet header	Apache attack	HTTP packets with large headers
Total similarity score from string comparisons in all the fields of pair wise packets	Apache attack	Multiple HTTP packets requesting same file
	UDP storm	Identical packets sent/received by victim machines
The access ratio of common files to uncommon files	Apache attack	Multiple HTTP packets requesting same file
Intensity ratio of incoming packets to outgoing packets	Apache attack	More HTTP requests arrive than are serviced
Access ratio of common to uncommon ports	Nmap scanner	TCP SYN packets received by all ports on the victim
	Netbus Trojan attack	Attempt to connect to the new port on victim
	Smurf attack	Increase in ICMP echo reply packets to victim
	IRC chat server abuse	New port opened for service on server
Same change in specific performance object on two/more hosts on the network	UDP storm	Increased network traffic by constant amount on both machines
Frequency ratio of common entries to uncommon entries in the Windows system/ security/ application log	Dictionary attack	Multiple login failures
Ratio of incoming to outgoing traffic volume per second	Meteor FTP attack	Fall in network traffic to ftp server
	Smurf attack	Increased incoming traffic
String in system/ security/ application log indicating start/ end of host /network application without normal procedure	Netbus Trojan attack	New system service added to startup automatically
	Half life attack	Client application terminated abnormally
File from restricted directory	EZPublish	File from settings directory of

accessed by non administrative user		EZPublish application is accessed
File with system specific extension accessed by non administrative user	EZPublish	File with system specific extension is accessed by non administrative user
Host connection to untrusted remote machine over the network	Half life attack	Half life client connects with untrusted half life server on the network
Network data with code to open a new shell on the system	Half life attack	Response has code to open a new shell
Multiple processes started by same parent process	Process table attack	Multiple calculator processes started by same user process
Registry key edited by user without administrative privileges	Yaga user to root attack	Unauthorized registry edit/New file added
User added/removed to system by process other than *usrmgr service*	Yaga user to root attack	New user added, not by usrmgr service
	Yaga user to root attack	user removed, not by usrmgr service
Sudden failure seen in using a system resource like network, files and memory	ARP poison	Victim is unable to reach destination

In Table 22, the first column generalizes the element that we monitor to detect an observable point of one or more attacks. The second column shows which attack profiles include this element, and the third column gives the attack specific observation for the element. By generalizing in this way, we can identify general attack elements that occur and investigate their frequencies to aid in developing sensors that can detect novel attacks.

4.3 Signal Detection Models

Once we have identified the DFC of a cyber attack signal, we need a model to detect it. To this aim, we leverage well-established signal detection models in the physical space (e.g., radar signal detection), and verify them in the cyber space.

This section proceeds as follows. We first present a literature review of signal detection in the physical space. We then summarize the feature extraction methods and detection models discovered in our literature review. The section concludes with a subsection mapping physical space, signal detection concepts to cyber signal detection in the form of DFC generalization and analytical discovery through data mining and analysis techniques.

4.3.1 Physical Space Literature Review

We survey existing signal detection methods in the physical space to create a collection of signal detection models. The search focuses on literature published in the years 1995 to 2004 (some literatures are unpublished). We find 163 papers from six different fields. This section describes the literature search and definitions of DFC and signal detection model used in this

study. The search includes six application areas and a theory area. Table 23 below summarizes the number of papers we review in each area.

Table 23. Number of papers review in each area

Area	Total
Biomedical & Health science	31
Earthquake & Planetary science	25
Economics, Finance, and Marketing Research	22
Manufacturing and Quality Control	28
Physical Signal Processing	39
Theory	28
Total	**173**

We conduct the search using various keywords depending on the area. Table 24 gives examples of the keywords used for each area.

Table 24. Summary of keywords used in literature search

Area	Key words used
Biomedical & Health science	Feature Extraction, Detection, Feature Selection, Hypothesis Testing, Noise Filtering, Pattern Recognition, Signal Detection, Signal Processing
Earthquake & Planetary science	Detection & Statistical, Detection, Fault Detection, Signal Detection
Economics, Finance, and Marketing Research	Detection & Statistical, Signal Extraction
Manufacturing and Quality Control	Fault Detection & Fault Signature, Fault Signature
Physical Signal Processing	Feature Detection & Statistical, Feature Extraction, Signal Detection, Detection & Statistical, Signal Extraction, Detection, Radar & Signal Detection, Radar & Statistical, Signal filtering, Data denoising
Theory	Data Fusion, Feature Extraction, Signal Detection

We find that economics, finance, and marketing research focuses on time-series data, with the goal of estimating (or extracting) the signal. Hence, most papers in this area only discuss the estimation method, and hardly mention the testing method.

Upon reviewing each paper, we extract the DFCs and signal detection model. Most papers contain all four elements but some provide only the DFC elements. We define these four elements below in the context of our literature search. These definitions reiterate and expand the definitions we give for attack profiling.

We define data as any variable of interest. For example, the data for detecting an underground water signal is the underground water sound. Data for each signal detection model

and attack type can be different. A special structure, form, or other attraction of the data is a feature of the data. We can extract features via data transformation. Examples of feature range from simple transformations such as raw data itself (no transformation required), mean, variance, distribution function, any linear combination, or ratio, to more complex transformations such as time series model, principal component function, Fourier transforms, or wavelet transforms. Each feature has a unique characteristic, which is useful in detecting a signal. For instance, characteristics of the mean can be a step or steady change. The characteristics that we wish to detect in data may not be the same as characteristic of feature. For example, step change in correlation coefficient implies a change in the data trend. Once DFC is defined a signal detection model can be designed to detect a particular signal. Generally, a signal detection model can be thought of as a process of making a decision – deciding whether or not the signal is present. Table 25 gives an illustration of mapping physical signal detection into DFC and signal detection model. Note that the characteristic shown in this table is a characteristic of the feature. Thus the signal detection model is designed to detect the feature characteristic [25].

Table 25. An example mapping physical signal detection into DFC and model.

Data	***Feature***	***Characteristics***	***Signal Detection Model***
Variable(s) of Interest EX Underwater sound signal	Vector of summary measures of selected wavelet coefficients EX Vector of mean sum of squares corresponding to high frequencies.	Step change in summary measures of selected wavelet coefficients	An adaptive model using recursive kernel estimation of joint distribution of sum of squares of wavelet coefficients. Any outliers from this kernel estimation are flagged as signal.

4.3.2 Physical Space Signal Detection Models

In the physical space, we find that the steps in detecting a signal are as follows. Firstly, data are processed by normalizing, scaling, or screening. Next the processed data are transformed to a feature. The transformation and feature are selected such that the signal is amplified. Hence the desired characteristic of signal can be detected easier and faster. Finally, a signal detection model is applied to the feature extracted from the processed data. Figure 15 illustrates these steps.

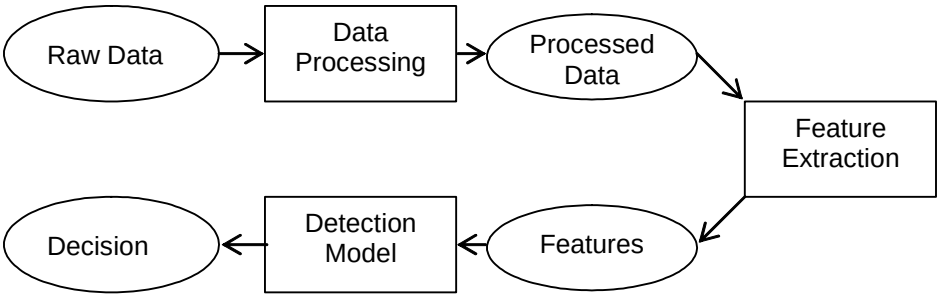


Figure 15. Common procedures in signal detection

There are many transformation methods available. Each transformation method is appropriate for a different kind of data and results in a different type of feature. Based on the literature search, transformation methods can be grouped into six categories: no transformation, simple transformation, regression based transformation, multivariate technique, transformation to frequency domain, and statistical learning method. Table 26 summarizes these transformation methods, features, and characteristics.

Table 26. List of all transformation methods, correspondent features, and characteristic.

Type of Transformation	Feature	Characteristic of Signal that we wish to detect
1.) None	Data itself	Any change from the template
		Any change in the distribution
		Change in absorption rate
		Change in causality relationship
		Change in clusters of data
		Change in correlation pattern in the data
		Change in data sequence
		Change in distribution (including any parameter change)
		Change in distribution parameter (e.g. mean, variance, scale parameter, etc.)
		Change in magnitude of signal
		Change in mean or change in pattern (i.e. change in slope or signal waveform)
		Change in parameter of autoregressive model
		Change in parameter of IMA model
		Change in spatial correlation pattern in the data
		Change in trend
		Change of slope
		Intermittent Sine Wave in the data sequence
		Single spike in the data sequence
		Step change in likelihood function

		Step change in membership function
		Step change in probability number of the presence of signal
		Step change in probability of the presence of signal
		Step change in belief mass function
		Step change in count of events (distribution mean)
		Step change in distribution mean
		Step change in mean
		Step change in parameter of interest
		Step change in probability of success
		Step change in time between events (distribution mean)
2.) Simple Transform-ation	Absolute difference	Any change in the data
	Autocorrelation function (ACF)	Change in autocorrelation pattern in the data
	Coefficient from steerable pyramid	Change in grey level of image data
	Conditional intensity	Change in conditional intensity
	Correlation coefficient	Change in correlation pattern in the data
	Correlation function	Change in correlation pattern in the data
	Data itself	
	Descriptor	Change in clusters of data
	Filtered data from Sum-box technique	Change in filtered data
	First nine moments	Change in clusters of data
	Four different weight sums	Change in grey level of image data
	Geometric mean	Step change in mean
	Inovention	Change in process state
	Kendall Statistic	Change in trend
	Knox Statistic	Steady change on mean of Poisson distribution
	Likelihood value	Change in pdf.
	Low- and high-pass	Change in clusters of data
	Mahalanobis Distance	Change from normal behavior
		Change in clusters of data

	Maximum Likelihood Estimate of signal shape and amplitude	Change in shape or amplitude of the signal
	Mean	Change in trend
		Step change in mean
	Mean and Variance	Change in location or spread of the data
	Mean of smoothed data	Step change in mean
	Median	Change in median
	Normalization of Low- and high-pass	Change in clusters of data
	Normalization of Time difference and integration of data	Change in clusters of data
	Normalization of high order statistic	Change in clusters of data
	Normalized correlation function	Change in correlation pattern in the data
	Numbers of right and left triples	Change in cycle pattern in data
	Sequential Probability Ratio Test (SPRT)	Step change in mean of input or output
	Signal to noise ratio	Change in magnitude of signal relative to noise
	Spearman Correlation coefficient	Change in trend
	Tango statistic	Change in clusters of data
	Third and four order statistic	Change in skewness and peakiness of data (or change in distribution shape)
	Third cumulant	Change in the skewness in data
	Third order statistic	Any change to non-linear signal
	Time difference and integration of data	Change in clusters of data
	Transformed data	Change in grey level of image data

3.) Regression-based method	Amplitude modulation estimate	Spike in the output mean
		Steady change in output mean
		Step change in the output mean
	Differencing-stationary parameter	Any change in seasonality parameter
	Fitted value from time series model	Change in underlying model
	Global factor estimate	Steady change in mean
	Margin of error of slope	Change in trend
	Predicted value	Steady change in mean
	Prediction error	Change in underlying model
	Regression coefficient	Change in correlation pattern in the data
		Change in underlying distribution
		Steady change in mean
	Trend estimate	Steady change in mean
	Trend or Cycle estimate	Steady change in mean
	Weighted sum of prediction error	Change in underlying model
4.) Multi-variate Method	Ai(k) index	Change in subspace spanned by the first m PCs
	Contribution of Principal Component (PC)	Change in correlation pattern in the data
	Dissimilarity index	Change in correlation pattern in the data
	Geometric vector	Change in correlation pattern in the data
	Independent Component	Change in correlation pattern in the data
	Latent Variable	Change in correlation pattern in the data
	Multiple Correlation Coefficient (MC)	Change in correlation pattern in the data
	Principal Component (PC)	Change in correlation pattern in the data
	Q statistic or Square Prediction Error	Change in correlation pattern in the data

	Reduced fault signatures	Change in clusters of data
	Residual	Change in correlation pattern in the data
	Singular value	Change in clusters of data
	T-sq	Change in correlation pattern in the data
	T-sq and Q-statistic	Change in correlation pattern in the data
5.) Transform to Frequency domain	Coefficient of periodic signal	Change in cycle pattern in data
	Expansion coefficient	Change in underlying model
	Harr coefficients	Change in data frequency
	Holder's exponent	Change in mean
	Moving average	Change in mean
	Principal component of wavelet coefficient	Change in correlation pattern in the data
	Probability density of residual	Change in mean
	Quotients which are the short- and long-term averages	Change in mean
	Singular exponents	Change in mean
	Sum square of error	Any change in high frequency band
	Time-averaged wavelet spectrum	Change in mean
	Transformed data	Change in mean
		Change in mean of transformed data
		Change in number of signals
	Wavelet coefficients	Change in data frequency
		Change in data mean
		Change in mean
	Wavelet packet node	Change in mean
6.) Statistical Learning method	Selected variables	Change in correlation pattern in the data
	Total Avidity	Change in clusters of data

While there are six categories of transformation methods, there are only two detection models. The detection models are threshold method and rule based method. A discussion of transformation method and detection model is given below.

Transformation method or Feature Extraction

No transformation use data itself (or processed data) as the feature and input to the signal detection model.

Simple transformation includes averaging. moving averaging, transforming to variance, higher order statistics, or correlation coefficient, filtering technique, and any simple transformation. Output from transformation is considered the feature. For example, features from simple transformation include mean, moving average, (Pearson) correlation coefficient, third or fourth order statistic, or filtered data.

Regression based transformation is used when the data consists of independent (x) and dependent variables (y). Common used regression based methods are least square model (OLS), generalized least square model (GLS), generalized linear model (GLM), spline regression, LOESS or non-parametric regression. This category also includes time series model such as autoregressive model (AR), autoregressive moving average (ARMA), or autoregressive integrated moving average (ARIMA). Regression based transformation can extract fitted (or predicted) value, residual, regression coefficients.

Multivariate technique is defined as transformation technique that considers all variables of interest simultaneously. Principal component analysis (PCA), independent component analysis (ICA), and partial least square (PLS) are grouped in this category. PCA and ICA are generally used when there is no dependent variable (y) while PLS is used with the presence of dependent variable (y). Possible features from these techniques are principal components, loading, and reduced data. Note that when reduced data is used, residual is another potential feature. When PLS is used the potential features also include those of regression based transformation.

Transformation to frequency domain usually appropriate when data have time series pattern or time domain do not reveal any useful information. It is often used in economics, finance, and marketing research. Methods include Fourier transform, wavelet transform, and Laplace transform. Potential features from this transformation are transformed data, reduced data, transform coefficients, and residuals (if reduced data is used).

Statistical learning method acts as transformation and signal detection in one algorithm. In general, data are used as an input to the model and the output from the algorithm is the decision. This method is computer intensive and involves in adjusting algorithm parameters. Hence it requires training data for fine tuning the detection model. Commonly used statistical learning methods are neural network, immune-81, and genetic algorithm.

Signal detection model

1. Threshold method consists of three classes: classical approach, Bayesian approach, and statistical learning method. The basic concept is to threshold on the feature or test statistic. If the feature value or test statistic is beyond the threshold value then the model signals an alarm. Usually threshold value is set to reduce the false alarm rate (or probability of type I error).

a. *Classical approach* is the conventional statistical hypothesis testing. The result from hypothesis testing is either yes or no. Examples of classical approach include t-test, F-test, ANOVA, (general) likelihood ratio test, χ^2 test, discrimination analysis, and change point detection. An adaptive version of these methods is considered as the classical approach as well. The adaptive method is used when the threshold value or calculation of test statistic is regularly updated using new observed values.
b. *Bayesian approach* can result more than yes or no type of answer. The concept is the same as classical approach but, instead of assuming the underlying model, Bayesian approach assume underlying conditional probability. Thus the threshold value is usually a function of conditional probability. Examples of Bayesian approach are Bayesian detector, Dempster-Schafer, Fuzzy algorithm, and association rule based on prior algorithm.
c. *Statistical Learning method*, like Bayesian approach, can result more than yes or no type of answer. Threshold value is in function of some kinds of similarity measure (i.e., error function, distance function, entropy, or loss function). Threshold value is usually set to either minimize false alarm rate or maximize detection rate or a combination of both. Hence statistical learning methods usually have an objective function. The threshold value is the value that will maximize that objective function.

2. Rule based method is simply a combination of threshold method. It involves "if then". For instance, a detection model alerts if number of alarm is greater than some threshold value where the alarm occurs when some test statistic (or feature) is greater than another threshold value.

4.3.3 Mapping Physical Space to Cyber Space

This section describes how we use aspects of signal detection from the physical space in designing a model for cyber space signal detection. Table 25 illustrates three DFC elements along with a signal detection model. To map this table to the cyber space, consider raw data (e.g., network traffic data) collected from computers and networks. The raw data goes through processing to obtain the desired data (e.g., the intensity ratio of packets for the web server to all packets) from which the feature is extracted using a feature extraction method (e.g., an arithmetic calculation of the sample average) [21]. Along with DFC, a corresponding signal detection model incorporates the characteristics of both cyber signal and noise, and monitors the feature to detect characteristics and decide if a cyber signal is present [21]. Table 27 illustrates an example of DFC and associated signal detection model in the physical space for the radar detection of a hostile object in the air, and in cyber space for the detection of the DoS attack from our previous example.

Table 27. Examples of DFCs and signal detection models [21]

Element	Physical Space	Cyber Space
Data	Radar image data	Packet intensity ratio
Feature	Shape & size of an object	Sample average (mean)
Characteristic	Shape is square & size is large	Step change
Signal Detection Model	A rule-based model: if shape is square & size is large, then signal	Cuscore model for step change

If we consider attack data as a signal to detect, and normal use data as noise mixed in with the signal in cyber space, then there is a mapping between cyber attack detection and signal detection in the physical space (e.g. radar and sound signal detection). Table 27 uses the cuscore model for step change to detect a signal in cyber space. Unlike existing techniques for cyber attack detection that rely on the model of only one element (signal or noise) in the monitored data, existing techniques for signal detection in the physical space often employ models that incorporate characteristics of both signal and noise, that is, all elements that exist and are mixed together in the monitored data [4,25-29].

We use the cuscore model to detect a step change in random noise that fluctuates around the level of T. The following noise and signal models are considered [4]:

Noise model: $y_t = T + a_{t0}$ (1)

Signal model: $y_t = T + \delta + a_t$ (2)

where T is the target value, a_{t0} and a_t are white noise and δ is the signal. The cuscore statistic is then defined as [4]:

$$Q = \sum_t a_{t0} r_t = \sum_t (y_t - T)\frac{(a_{t0} - a_t)}{\delta} = \sum_t (y_t - T)\frac{\delta}{\delta} = \sum_t (y_t - T). \quad (3)$$

where r_t is the Detector (rate of change of background noise).

This cuscore model is sensitive to detecting a step change signal buried in random noise. For any random variable, after we take account of the first part we will always have the second part of random white noise to account for the randomness of the variable. The second part is the standard form of mathematical modeling. Box and Luceno provide other cuscore models that are constructed to detect: a sine wave, slope change and single spike signal buried in the random noise of equation (1), and parameter change signals with the noise of a first-order autoregressive time series model or the nonstationary disturbance noise of an (IMA) time series model [4]. Many signal detection techniques in the physical space, including low-pass and high-pass filters, use frequency bands to characterize and differentiate signal and noise to perform signal filtering or detection accordingly [27].

The attack-norm separation approach consists of the following three steps in order to detect an attack:

1) Define models of cyber signal and noise
2) Filter out noise from mixed data, using the cyber noise model
3) Identify the cyber signal in the remaining data, using the cyber signal model

For example, in the cuscore model equations (1) and (2) carry out Step 1 of the attack-norm separation approach by defining the noise model and the signal model. The signal model

indicates that the step change signal is added to the noise. Hence, it is an additive signal model. Note that not all signals are additive. Some signals may distort the noise in other ways than simply adding a signal to the noise. Steps 2 and 3 of the attack-norm separation are embedded in equation (3)

Previously we described how we generalize observations across attacks. Here, we extend that to identify detailed aspects of the generalized cyber attack variables, including data, feature, transformation method for feature extraction, characteristic of feature and signal detection model. Table 28 gives an example of some data generalizations.

Table 28. Examples of generalized data from existing attack profiles

Type of Data	Data	Feature	Transformation Method	Characteristic of Feature	Signal Detection Model
Single Source	Raw data: header fields of each packet Computed variable: total similarity score from comparisons of all fields between consecutive packets	No transformation	N/A	Step change	1) t-test using observations over time 2) cuscore for step change
Single Source	Raw data: a string indicating the start or termination of a host or network application (e.g., FTP, www) or user (e.g., system administrator) in the Windows registry, security, system and application log	No transformation	N/A	Special string value	String match
Multiple Source *Intensity Measures*	Raw data: a variable measuring incoming and outgoing traffic volume per second	No transformation	N/A	Step change	1) t-test using observations over time 2) cuscore for step change
Multiple Source *Intensity Measures*	Raw data: a variable measuring incoming and outgoing traffic volume per second	Wavelet coefficients	Haar and Complex Wavelet analysis	Multivariate pattern difference between signal and noise	1) ANOVA 2) hierarchical cluster 3) decision trees 4) a table of wavelet coefficient vectors, all using vectors of wavelet coefficients from multiple signal and noise sessions

The first column in Table 28 groups the variables based on data source and measure of interest. Data sources here are single or multiple. For multiple data sources, we can look at different measures, such as intensity measures (time-driven), activity pattern measures (event

driven) and activity-state-performance interaction measures. The examples in Table 28 cover single source and multiple source w/intensity measure data. The remaining columns give examples from attacks of data, feature, transformation method, characteristic and signal detection model.

Considering Figure 15, step 1 takes raw data, step 2 processes the data (centered, scaled, screened, etc.) to obtain the variable in step 3 (processed data). Steps 2 and 3 are optional depending on whether or not the variable of interest requires pre-processing of the raw data. In step 4 we extract a feature by transforming the data variable using one of the feature extraction methods given in Table 26 to obtain the desired feature in step 5. At this point, we need to use a detection model which looks for a certain characteristic on the feature. In step 6 we consider one of the signal detection models given previously. Finally, in step 7 we make a decision based on the outcome of step 6. Figure 16 gives an example of using this 7 step process to detect a signal in the physical space. This example does not require the optional stages of transforming raw data into desired variable. Thus, Figure 16 shows the details for steps 1 and 4-7 for this example.

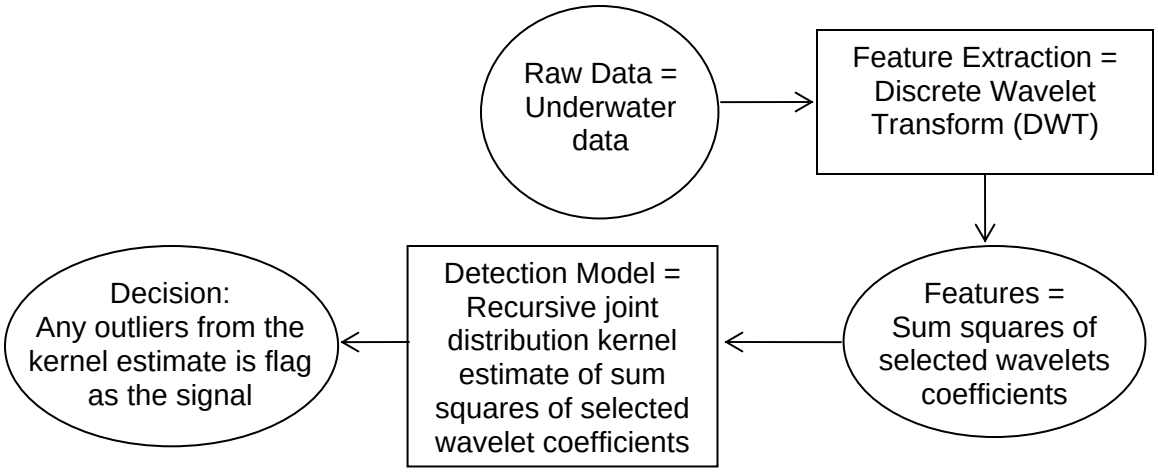


Figure 16. Example of the steps in detecting a signal in the physical space

We can use the same procedure for detecting a signal in cyber space. Table 29 gives a sample of commonly used feature extraction methods from the physical domain that we can consider for use in the cyber space. This list is a created from Table 26 and extends each row to include the corresponding data type, underlying model and signal detection model.

Table 29. Commonly used feature extraction methods from the physical domain

	Data Type	Underlying Model	Feature(s)	Transformation Method	Characersitic of Feature	Signal Detection Model
Simple Transformation	Any	N/A	Correlation coefficient	Pearson's correlation coefficient	Step change	MAROC
		N/A	Mahalanobis Distance	Mahalanobis Distance	Step change	Linear discrimination model
		N/A				Multi dimensional heuristic method
		N/A				Quadratic discrimination model
		N/A	Mean	Sample average	Step change	T-test
Regression based method	Any	y = b0 + b1x	Trend estiamte	Least squares regression	Step change	Mann-Kendall Statistic
						Mann-Whitney Statistic
						T-test
	Input and output	EWMA: Zs = lSum(1-l)s-tX(t)	Fitted value	Time series model (not specified)	Step change	Prediction interval
		logit[p(y)] = -c + dx	Regression coefficient	Genearlized Linear Model	Step change	T-test
		OLS model: Yj = a + bj + ej	Margin of error of slope	Least squares regression	Step change	T-test
		Seasonal Model Yijk = m + Ti + Mj + eijk	Regression coefficient		Step change	T-test
		Y = Xb + e	Regression coefficient	Any regression technique	Step change	T-test
	Time series	jh(B)(ht - m) = qh(B)bt	Regression coefficient	ARMA model	Step change	T-test
		f(x,w) = w y(x) + b	Fitted value	Adaptive support vector regression	Step change	Confidence interval on predicted value
		Y = FQ + Y	Prediction error	Bayesian Regression	Step change	Confidence interval on predicted error

Our approach includes 3 basic stages:

1. Identifying specific data to collect corresponding to the specified raw data derived from attack profiles. This stage includes defining the specific data sources. Table 30 illustrates the identification process

Table 30. Defining variables for associated data sources

Type of data	Data	Identified source of data
Single source	Raw data: header fields of each packet	IP packets – source address/destination address packets, port number
Multiple source Intensity measure	Raw data: a variable measuring incoming and outgoing traffic volume per second	IP packets per second

2. Extracting data in the log files collected from each attack using specific programming tools and statistical packages. The identified data from each log file need to be transformed into an analyzable form (variable) to run in the tests defined in the analysis stage and to extract some useful information from them. Table 31 illustrates the extraction process:

Table 31. Extracting variables

Identified source of data	Extracted variable for analysis
TCP port number	Similarity score of port numbers by comparing consecutive packets
IP packets per second	IP packets per second from the performance log file extracted using Statistica.

3. Analyzing the extracted variables. Two methods are defined for analyzing the variables (Table 32):
   - Directly analyzing the extracted variable without any transformation using T Test to find out the step change in data
   - Transforming the extracted variable using wavelet transform and finding the pattern change in data between signal and noise. Haar and Morlet wavelet transformation methods are used for this.

Table 32. Analyzing extracted variables

Extracted variable for analysis	Transformation method	Signal detection model
Similarity score of port numbers by comparing consecutive packets	No transformation	T Test using observations over time
IP packets per second from performance log file	Both no transformation and wavelet transform	T Test, ANOVA, a table of wavelet coefficients

The verification process is outlined in Figure 17. In this figure, step 1 is the raw data, step 2 corresponds to stage 1 above, step 3 corresponds to stage 2 above, and steps 4 and 5 correspond to stage 3 above. In this way, we are able to verify the physical signal detection models for detecting cyber signals.

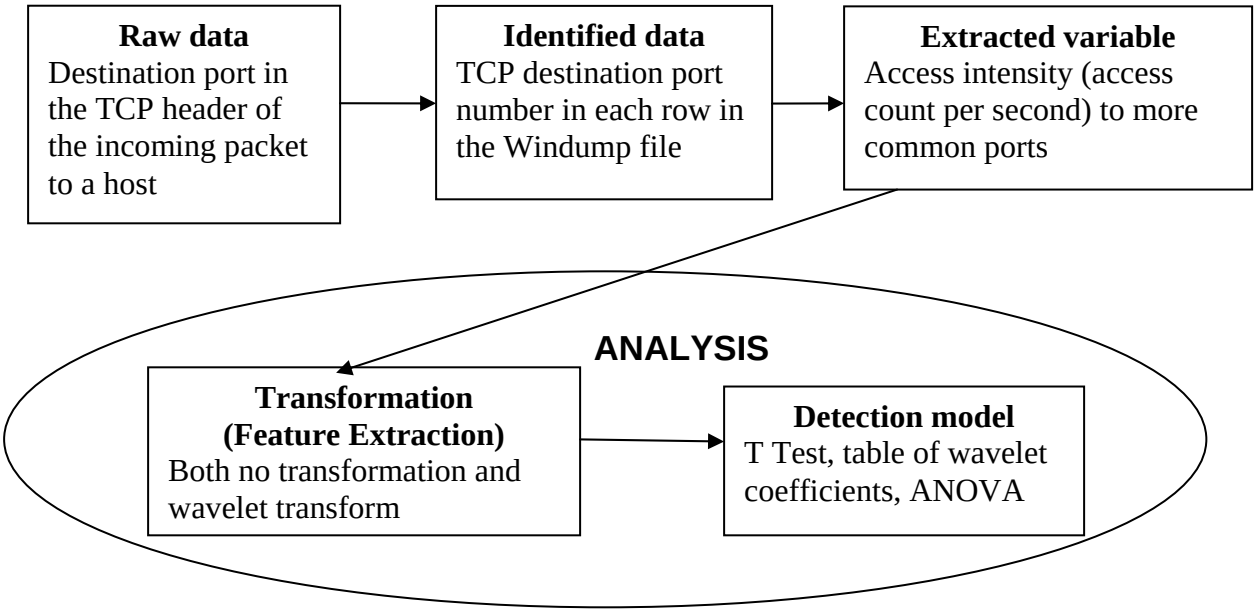


Figure 17. Example of verification process

4.4 Attack Simulation and Data Collection

This section explains attack simulation and associated data collection carried out in the project. It is divided into three subsections. The first explains concepts that are common across all attack simulations. The second explains attack-specific information on simulation and data collection. The third describes the worm simulation. This section is a result of the simulation of selected attacks and worm, and covers all relevant implementation details.

4.4.1 Setup Common to Attacks

This section explains concepts common across all the attack simulations. For each attack simulated, we collected data from various sources on the hosts as well as the network. Table 33 describes the various sources for data and the tools used to collect them.

Table 33. Data collected and collection tools

Data collected	Tool used	Location
Event logs (System, security, application)	Windows Event Viewer	attacker, victim, bystander
Performance logs (all counters available)	Performance monitor utility	attacker, victim, bystander
Registry logs (all registry accesses on the host)	regmon utility	attacker, victim, bystander
Network data logs (headers of all packets on the network)	windump utility	Router

The windows event viewer tool allows access to three event logs: the system, security, and application log. It can be cleared or saved at any given time. In order to select all audit events, we need to initially select options in the security policy. The following procedure is needed only once in a new machine, and can be used for subsequent simulation runs.

1. Control panel -> administrative tools -> Local Security Policy.
2. Open “Local Policy Folder”
3. Click on “Audit Policy” Folder
4. For each item in the “Audit Policy” folder double click on it, and when the “Local Security Policy Setting” window comes up, select both success and failure check boxes, followed by “OK”.
5. You may need to close and re-open “Local Security Policy” or restart before the “effective” policy changes.
6. Find and right click on the “WINNT” folder in “My Computer”.
7. Select “Properties”.
8. Select the Security Tab of the “WINNT Properties” window that pops up.
9. Click the “Advanced…” button.
10. Select the “Auditing” tab of the “Access Control Settings for WINNT” pop up window.
11. Click the “Add…” button.
12. Select “Everyone” from the “Select User or Group” pop up, and click “OK”. The “Auditing Entry for WINNT” window should pop up. (You can also reach this window by clicking on the “View/Edit” button on the “Access Control Settings for WINNT”).
13. in the “Auditing Entry for WINNT” select successful and failed check boxes for all the access types. Make sure that Apply onto is set to “This folder, subfolders and files”. The box marked “Apply these auditing entries to objects and/or containers within this container only” should be unchecked.
14. Select “OK” on the “Auditing Entry for WINNT” window.

15. Select “OK” on the “Access Control Settings for WINNT” window.
16. Select “OK” on the “WINNT Properties” window.

The performance monitoring utility can be turned on locally or remotely using the performance tool. All performance objects were selected and collected once every second (default is once per 15 seconds). Both of these tools are accessible from *Control Panel -> admin tools*.

To capture registry activity, a utility called regmon is used [30]. It records all accesses to the windows registry in real-time, and is a freely downloadable utility. Its output can be saved to a text file directly.

To capture network packets, a utility called win dump is used. Win dump is a windows port of the famous tcpdump utility [31]. Specifically for our setup, we used the following command line: **windump –I 4 –w packlog.bin**

For most attacks, the following setup is used:

- Network of 5 machines each having similar configuration.
- Logger machine is used as a remote performance logger.
- Clock server used to synchronize clocks before each attack simulation.
- Gateway is used to collect network data during the attack. In this setup the router has only one host connected, and thus will only route packets received for that host.
- The other 3 machines are labeled Attacker, Victim, and Bystander.

Figure 18 describes the simple setup used for the initial simulations of these attacks. All the machines are connected to a common hub. For our initial simulation, data collection and discovery, we only considered data at the victim (due to time constraints and the enormous amount of data collected). This simple setup is sufficient to explore many variables which are unaffected by whether or not the attacker is on the same subnet, thus is representative of a realistic environment with respect to those variables.

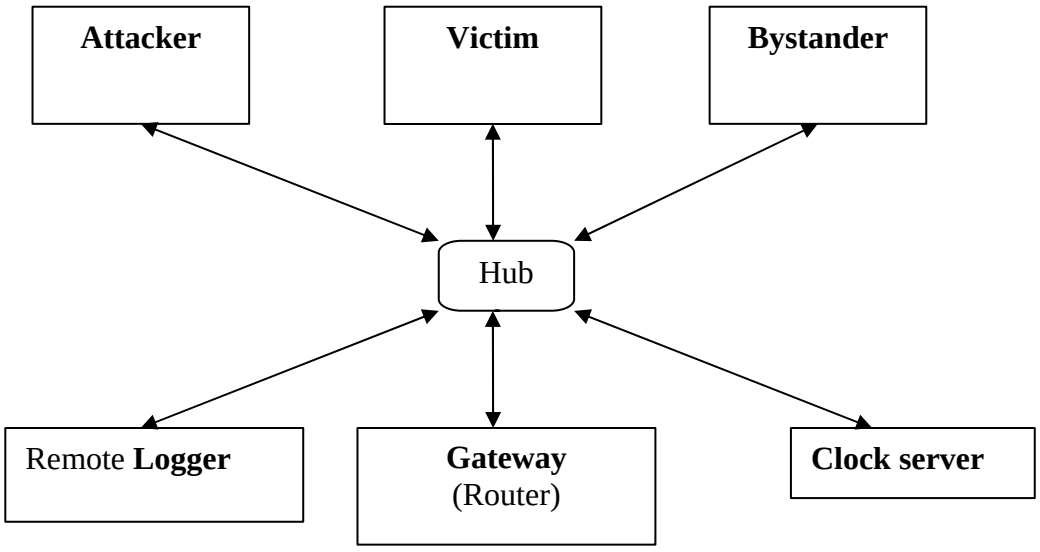


Figure 18. Setup used for simulation of attacks

In order to compare normal scenarios with attack scenarios, each attack simulation is run in three phases. Simulation of attacks is done such that normal phases occur before and after the attack phase. This allows the study of pre attack normal data and after-effect data along with attack data. The phases are-

- Normal activity data (First 10 minutes), called the normal phase.
- Attack data (variable time period)
- After-effects data (10 minutes)

Attack phase time varies across attacks. For example, in the Meteor FTP attack, this is almost instantaneous with a single input packet from the attacker, whereas with the Apache attack, the duration of attack is chosen to be 10 minutes. Figure 19 describes the three phases of simulation/data collection, and their timelines.

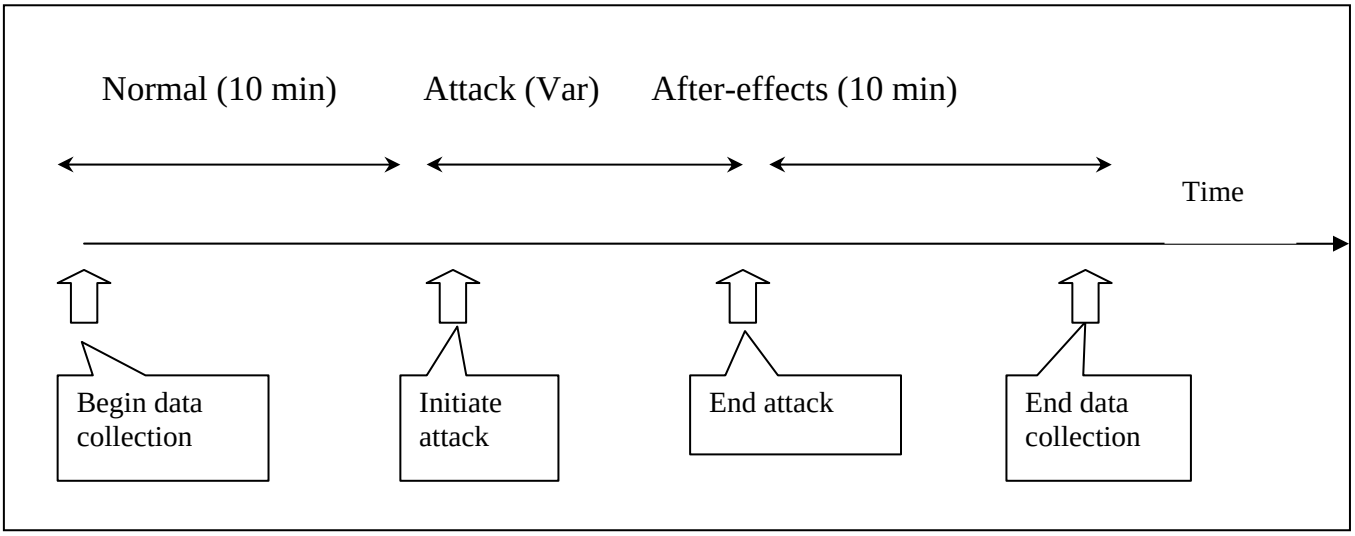


Figure 19. Data Collection Scenarios – Local and Remote

Logging all performance counters once per second creates huge log files, which get written to disk during the simulation. Writing such a huge file to the disk/storing it in memory affects the performance counters. Thus, the data logs reflect two effects –

1: Effect of attack simulation

2: Effect of data collection

Since it is not possible to get rid of this effect of data collection, an alternative approach is used. Performance counters for all machines (attacker, victim, bystander) are collected on a remote machine. Thus, the effect of writing huge logs to the disk is no longer a problem. However, this leads to increase in network utilization and corresponding events in the system.

Thus, we have two modes of data collection – Local and Remote. The difference is only with the large performance logs, which are collected on the local machine itself in the former case and are collected remotely in the latter case.

Note: To enable remote performance logging, we need to allow access in *control panel ->services.*

The event, performance, registry and network data logs collected are in their raw state, and require some amount of post-processing. The following two steps need to be done.

For Windump, read raw log data from the binary file and convert to text mode, and save in another file. The command used for this is: *windump –r packlog.bin >> network_data.txt* Here *packlog.bin* is the original file and *network_data.txt* is the converted text file.

Separate the data logs into the three phases of normal phase, attack phase and post-attack phase. To achieve this, we need to identify the position in time in each log, at which the attack begins and ends. As discussed before, for most attacks, the attack phase is 10 minutes, but for a few attacks, the attack is a lot less than 10 minutes. These times are identified from analysis of each attack. Once these times are identified, we insert special strings in each log file, indicating the start and end of the attack. The strings used for this purpose are *<attack begins here>* and *<attack ends before here>*. The time is rounded to the next higher minute.

A small perl script is used to insert the strings in the performance logs. At this point, the filenames, the string to be inserted and the time at which they are inserted are part of the script itself. (In the future, they can be taken as parameters to the script.) For the other logs, insertion of the strings is fairly trivial and is done manually.

The following procedure is followed during simulation. Computer clocks for all five systems involved in the simulation are synchronized against the same time server

- Before each simulation, all event logs (application, system, and security logs) are cleared
- Windump packet logger on Gateway is started
- Performance logs are started (local or remote)
- Registry monitoring is turned on.
- Wait for 10 minutes of normal activity
- Initiate attack script/program
- Wait for 10 minutes/lesser time for attack to complete
- Allow 10 minutes of after effects time
- Stop performance counters
- Registry monitoring is turned off and logs saved
- Windump is stopped and log saved
- Performance logs are stopped and logs saved
- Event logs are saved.

4.4.2 Specific information for each attack

The following section explains attack specific simulation/data collection details, if any. If there is no change from the normal simulation described above, then the particular attack is listed without any explanation.

4.4.2.1 EZPublish Confidentiality Attack

EZPublish is software which has a confidentiality vulnerability [32]. A critical system file is not protected with the right set of permissions, allowing anyone to access it and read the underlying database's username/passwords and other information.

- Machines: Attacker, Victim, Bystander
- Attack: Attack includes opening a file on the remote machine, and saving it to local disk. All this is done in under 1 minute. Very likely that attack activity is similar to normal activity, since this is a confidentiality attack.
  - Victim has EZPublish software installed on the machine, and allows remote connections. Default settings are enabled.
  - Attacker machine connects through the LAN and issues request to the software on victim for the settings file.
  - The file is returned to the attacker, thus compromising the security.
- Simulation timelines: 20 minutes of normal data, followed by attack (under 1 minute), followed by 10 minutes of after effects. Here, the attack is a single request for a particular file, and the attack ends as soon as the file transfer is successful.

- Configuration: For this attack, the only configuration needed is to setup the EZPublish software on the victim, and query the victim with a request for the specific system file. This request would be: http://[target]/settings/site.ini, where target is the IP address of the victim machine.

4.4.2.2 Nmap Scanner

The Nmap scanner is used in this simulation for two purposes – scanning the ports for a list of all open ports on the system, and probing every open port to learn what service /version of software is running at the port [33].

- Machines: Attacker, Victim and Bystander
- Attack: The attack uses the standard nmap program, which can be freely downloaded from the Internet. Options in the nmap program allow selection of victim, selection of ports to be scanned, selection of options to probe for services at each open port and other options.
- Simulation timelines: 10 minutes of quiet/no attack, followed by approx. 5 minutes of nmap probing, followed by 10 minutes of quiet aftereffects. Since the probing program automatically terminates after 5 minutes, there is no need for manual intervention to stop the attack. Note: In the remote logging case, normal phase before attack is 13 minutes, instead of 10 minutes, due to an oversight.
- Configuration: nmap -P0 -p 1-1024 -v -v -sT -sV Victim. This scans ports 1 through 1024, gives verbal output, does a version scan and a stealthy scan on Victim machine. These were found to be open TCP ports on the victim: 7, 9, 13, 17, 19, 21, 135, 445

4.4.2.3 Netbus Trojan

Netbus Trojan is a Trojan program that gets installed when a legitimate program is installed by a user. The original installable itself is affected, so the user is not aware of the installation.

- Machines: Victim and Bystander
- Attack:
  - Install malicious program with the game on the victim.
  - Wait 5 minutes.
  - Connect from attacker's machine to victim's machine through netbus backdoor.
  - Do a screen dump of victim onto attacker machine as a proof of concept. This shows the contents of the victim's screen to the attacker.
  - Attacker has a installable zip file, for a poker game. The victim installs this file from the attacker machine, thinking that it is a game installable and installs it. When the game is installed, automatically the netbus server is also installed. This is achieved by a simple batch script. (install.bat)
- Simulation timelines: 10 minutes of normal activity, followed by attack phase for 5 minutes, followed by 10 minutes of after effects; Here, the attack phase includes-accessing the malicious executable program on the victim, installing the program (and inadvertently the backdoor) which occurs in less than a minute. This is followed by about 5 minutes of silence and then the attacker connects to the victim through the backdoor

and gets a screen dump of the victim's machine. Once the screen dump is complete, the attack is considered complete.

- Configuration: A small script is written as a batch file in MS-DOS. This installs a game of poker, along with the netbus Trojan server on the victim. The attacker has a netbus client, and uses the IP address of the victim to connect and exploit its vulnerability.

4.4.2.4 Meteor FTP

Meteor FTP is a popular FTP server that is available for download on the internet [34]. This software has buffer overflow vulnerability, in the username field. Thus, if a long username is supplied by the user, the application is unable to handle it and crashes.

- Machines: Attacker, Victim and Bystander
- Attack: Remotely connect to the FTP server. In place of the Username, enter USER followed by a set of random characters. The server will crash after spitting out an error message that unauthorized area in memory was being accessed
- Simulation timelines: Since this attack is active only when the username request is sent to the client and the client responds with a single long string, it takes less than a minute. Thus, this attack has 10 minutes of normal phase, followed by an attack phase of under one minute, followed by 10 minutes of after effects.
- Configuration: For this attack, the Meteor FTP server is installed on the victim machine. The attacker connects to the victim machine through the FTP service, and issues a long username as the input. This leads to crashing the service on the victim machine. The attack is fairly straightforward.

4.4.2.5 IRC Chat Server Abuse

IRC allows multiple users to login to the char server program, enter a specific chat room and chat with the other users. This is not an attack per se, it is more a misuse of computing resources and user time. Thus, there is no attacker, victim for this attack. Rather, there is the server and the client.

- Machines: For this simulation, the IRC chat server is setup on one machine, while the char client is setup on the other machine. Thus we have: Client, Server and Bystander.
- Attack: For this simulation, attack involves connecting from the client to the server, entering a chat room and waiting for 10 minutes with the connection between the client and server still on. As long as the connection is active, the server/client sends ping packets back and forth to verify the connection status.
- Simulation timelines: 10 minutes of normal session, followed by ten minutes of attack session, followed by ten more minutes of aftereffects.
- Configuration: The IRC chat server and client are downloadable from the Internet. The chat server is installed on the Server machine, while the client is installed on the Client machine. The client needs to be configured with the IP address/port of the chat server.

4.4.2.6 ARP Poison

Attacker sends malformed ARP update packets on the LAN, corresponding to the victim's IP address [8]. This makes the machines on the LAN learn the wrong MAC for the victim's IP address. Any attempt to reach the victim might fail if the machines learn the wrong MAC address. (This information is stored in the ARP table of the machine)

- Machines: Attacker, Victim and Bystander
- Attack: The attack executable/program used, sends a series of malformed ARP packets on the LAN at regular intervals, during the attack phase. Each packet sent provides the wrong MAC address to the IP address of the victim. Thus, the attack is considered on, as long as the attack program is run. The attack stops when the attack program is terminated (with a Control + C)
- Simulation timelines: 10 minutes of no attack, followed by attack initiation for 10 minutes, followed by after-effects for 10 minutes.
- Configuration: The attack program was written in C++, and compiled on VC++. The parameters like time between packets, IP address under attack and MAC address sent are configurable in the program itself. Any change to the configuration will require a recompilation of the program.

4.4.2.7 Sobig Worm

Sobig is an e-mail worm. It requires user intervention to spread - a user must be tricked into executing the attachment containing the worm. Once executed, the worm sends out mass e-mails to e-mail addresses found on the compromised machine.

- Machines: Attacker, Victim, Bystander and an additional machine Server to act as a local mail server.
- Attack: The attacker (an infected machine) sends an e-mail with the worm payload to the victim machine - the worm searches the victim machine for e-mail addresses and sends e-mails with the worm payload to additional machines, including the victim, through the local mail server. When a user on the victim machine opens the e-mail and is tricked into executing the attachment, the worm infects the victim machine as well.
- Simulation timelines: 10 minutes of no attack, followed by approximately 10 minutes of attack (where the worm searches for the victim's e-mail address and sends out infected e-mails), followed by after-effects for 10 minutes.
- Configuration: For this attack, an e-mail server must be installed and running on the local network. A DNS server must also be present. E-mail clients (e.g., Microsoft Outlook) must be present on both the attacker and victim machines. To start the attack, the worm is executed on the attacker machine. The attacker machine must contain a text file that possesses an e-mail address associated with the victim machine.

4.5 Analytical Discovery

To develop models of cyber attacks, we discover characteristics of cyber signal found in our simulation data. We verify these findings by comparing them with our profiles of cyber

attacks. We investigated many data analysis techniques, and found the techniques given in this section to be the most useful in this investigation. Our investigation included extracting features from raw attack simulation data and looking for distinguishing characteristics on those features. In this section we present some analysis results.

These results are only a sampling of the analysis we have conducted. Here we only present the analysis results from one of three or four machines for each attack simulation. The results presented here are greatly summarized as the actual amount of analysis we have done is quite extensive. In the next subsections for the six attacks simulated, we only show those variables that show any change between any phases and appear as such in every attack. The complete scope of our analysis shows the result of every variable for every stage in every attack. For the same reasons, we only interpret and discuss some of the results, within the scope of this project. Our future work and publications will give more details of our analytical discovery.

4.5.1 Correlation, Distribution and Difference in Mean

For this section, we include the following results for each analysis (autocorrelation, Pearson correlation, distribution, and difference in mean):

1. For each of the 6 attacks we compute the following 18 lists
    - All variables that change between pre-attack/attack
    - All variables that change between attack/post-attack
    - All variables that change between pre-attack/post-attack
2. We combine these list together for each attack (resulting in 6 lists)
3. We compare these 6 lists and pick out the common variables
4. We pick one variable that is in one of the 18 lists from step 1 and is not in the list from step 3. For this variable, we propose a suggestion as to why it *is not* common among attacks.
5. We pick one variable from the list in step 3. For this variable, we propose a suggestion as to why it *is* common among attacks.

This report gives the results of steps 3, 4 and 5. However, we do keep the lists from steps 1 and 2 for further study.

We first present the procedures we follow for data analysis. This includes an illustration of the entire scope of our data analysis, some of which is outside the scope of this report. Next we give results from four tests on six attacks: probability distribution, autocorrelation, Pearson correlation and difference in averages. Finally we present our results of the one worm covered in this section.

4.5.1.1 Procedures

This section outlines the data analysis tests we performed with the tools we developed to run these tests. This is a comprehensive list of the data analysis process for the performance object (performance log) data, and includes steps that we complete, but do not include results for in this report.

Steps in analyzing data: Performance Logs

For raw data file containing data from all three stages: pre-attack, attack and pos-attack

1. Filter out the first 10 observations, using program, and all-zero variables
2. Filter variable with zero variance, using program
3. Separate the file into three files for : pre-attack, attack and post-attack data , based on string inserted

For preattack OR attack OR post attack files

1. Filter out the all-zero variables
2. Replace missing data with mean, using program
3. Run basic stats: mean, variance, minimum, maximum, standard deviation, etc.
4. Filter variable with zero variance, using program
5. Run correlation matrix, histograms, hierarchical clustering of variables manually
6. Run percentage of significant correlations, using program
7. Scale data using ((x-mean)/SD) for autocorrelation
8. Fix long name variables problem manually
9. Run autocorrelation analysis, using program
10. Run KS Test, and Chi Square Test for distribution testing
11. Run Skewness and Kurtosis, using program (These have to be done after Step 1 of the Mixed files, using common-variables-filtered individual files)
12. Run ANOVA on percentage of significant correlations results from parametric method *For mixed files (pre-attack followed by attack*) *(attack followed by post- attack*) *(pre-attack followed by post-attack*)
13. Compare the two files to filter out the uncommon variables to keep only variables that exist in both normal and attack files, using program
14. Run time series plot (sampling every second), using program
15. Run t-test, and filtered only common variables regardless of data collecting method, and victim machines, using programs
16. Import files into Statistica Miner to run Decision Tree

Steps in analyzing data

1. Run Pearson, and Spearman for correlations coefficient between variables
2. Run ANOVA on percentage of significant correlations results from nonparametric method
3. Run Mann-Whitney U test, T-test for difference in averages

In this report, we only consider the numerical data from the performance logs (performance object data) generated during attack simulation and data collection. The data we analyze here is from one machine. In five of the attacks, this is the victim machine, for IRC Chat,

it is the chat server. We use the terms common and uncommon in two cases. With respect to local and remote data collection, common variables are those that appear with the same characteristics in both collection methods. This eliminates the possibility that the variable is effected by data collection method. For attacks, a common variable shows the same characteristic across all attacks in this study, whereas an uncommon variable does not. The analysis results we computed and collected are as follows:

Distribution Analysis

- Common variables between local and remote collecting that fall into each distribution for each phase
- Common variables between local and remote collecting that shift distributions between phases
- Common variables among all attack types that shift distributions between phase
- Uncommon variables among all attack types that shift distributions between phase
- Skewness and Kurtosis change between phases
- Variances Difference

Correlation Analysis

- Variables list that change significance of correlation coefficient between phases (common between local and remote)
- Variables list that are significant in each phases (common between local and remote)

Autocorrelation analysis

- Variables that are very uncorrelated for each phase (common between local and remote)
- Variables that are highly correlated for each phase (common between local and remote)
- Variables that shift autocorrelation between phases(common between local and remote)
- Common variables across 6 attacks
- Uncommon variables for each attack

Difference in mean

- Variables that shows difference in mean between phases
- Common variables that show difference across all attack type
- Uncommon variables that show difference for each attack

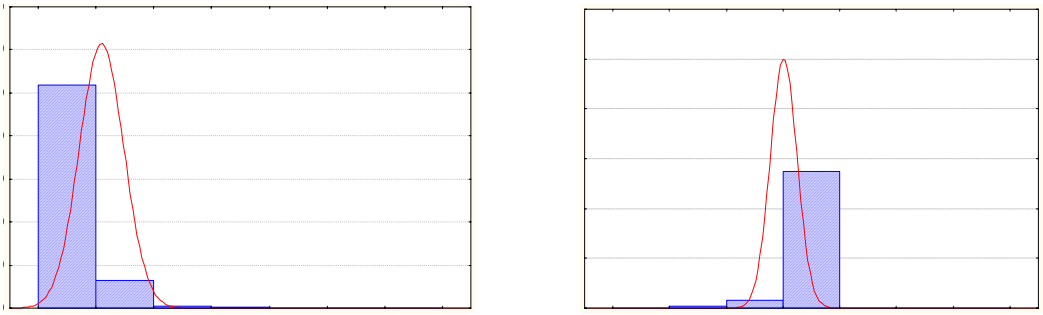
Although we collect and save many intermediate results, we only present here the higher level findings for the sake of brevity.

4.5.1.2 Six Attacks: Probability Distribution of Variables

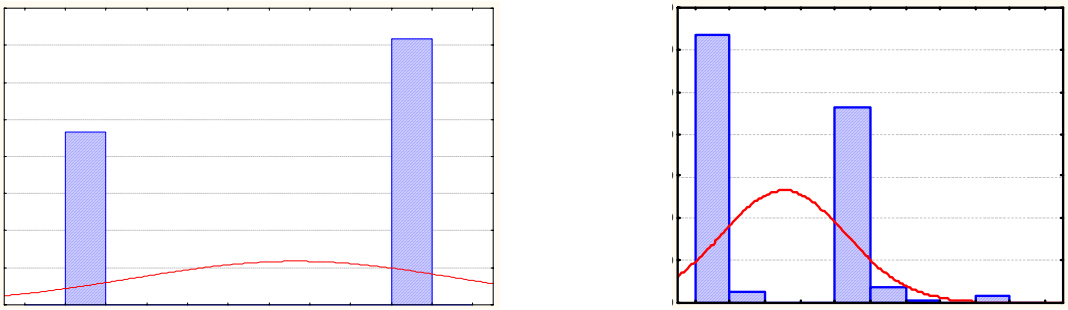
We experiment to identify the probability distribution of variables in each phase: pre-attack, attack and post-attack, and find differences in the distributions of variables among the three phases to use as the observables in identifying an attack. Due to the small size of some attack phase data, we use the Kolmogorov-Smirnov (KS) test for probability distribution because it is reliable even on small datasets.

Analysis results show that there are four types of distributions commonly found across all 6 attacks. Figure 20a-c shows an example of what each of these types of distributions look like:

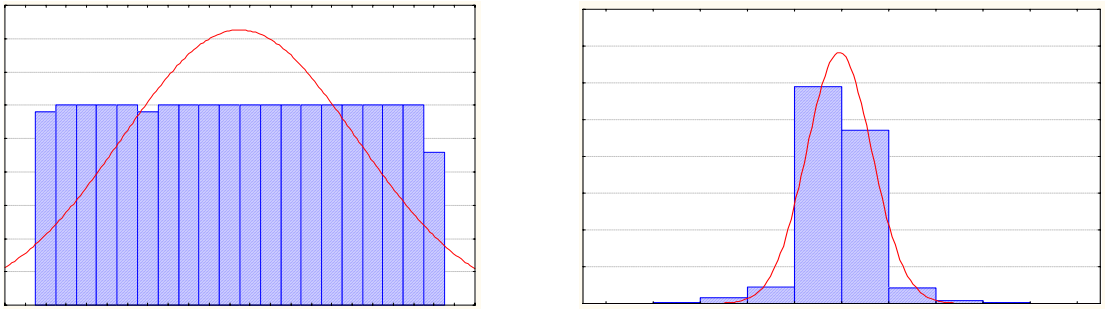
a) Skewed distribution
b) Bimodal distribution
c) Uniform and symmetric (potentially normal) distribution



a) Examples of right and left skewed distributions



b) Examples of bimodal distributions, consisting of two uniform right skewed distributions



c) Example of uniform and symmetric distributions

Figure 20. Example shapes of distributions

Other significant findings show the following:

- For all 6 attacks, the most common type of distribution found overall is right skewed
- There are many variables that shift to a normal or uniform distribution, from one phase to the next (pre-attack to attack, or attack to post-attack)

We conduct the KS test on all 6 attacks to test whether the variables fall into any of the three distributions: uniform, exponential, and normal. Due to the characteristics of our simulated data, we can only test exponential distribution (a right skewed distribution), uniform distribution, and normal distribution (a symmetric distribution). Also, currently there is no statistical test for testing bimodal distribution. As an example of how we derive distributions for the variables, Figure 21 shows the KS and Chi Squared test results from 3 machines for 3 probability distributions and 3 phases of data collection from 2 locations.

	KS			Chi Squared			Total
Distributions ---->	Uniform	Exponential	Normal	Uniform	Exponential	Normal	
Pre-attack Local							
Alpha01	35	0	0	35	0	0	560
Alpha02	37	0	3	37	0	2	449
Alpha03	39	0	1	39	0	0	398
Pre-attack Remote							
Alpha01	34	0	0	34	0	0	573
Alpha02	36	0	0	36	0	0	601
Alpha03	35	0	8	35	0	0	497
Attack Local							
Alpha01	43	0	48	43	1	48	455
Alpha02	49	0	60	49	9	59	332
Alpha03	61	0	76	62	1	73	262
Attack Remote							
Alpha01	34	0	33	34	5	33	527
Alpha02	49	0	82	51	0	51	374
Alpha03	64	0	64	63	0	61	230
Post-attack Local							
Alpha01	36	0	3	36	0	2	439
Alpha02	49	0	2	49	0	1	367
Alpha03	53	0	3	50	0	0	322
Post-attack Remote							
Alpha01	38	0	0	38	0	0	387
Alpha02	47	0	0	48	0	0	369

•This table shows that many variables are normally distributed only during the attack phase

Figure 21. KS and Chi Squared sample test results

Table 34 shows an example of the number of variables that fall into a particular distribution, using KS and Chi Square test from EZPublish attack.

Table 34. Number of variables that fall into a particular distribution in EZPublish attack

Phases	Distributions			Total Variables in Dataset
	Uniform	Exponential	Normal	
Pre-attack	35	0	0	601
Attack	48	0	54	374
Post-attack	50	0	0	369

From Table 34, we see that none of the variables fall into normal distribution in pre-attack and post attack phase, but 54 variables fall into normal distribution only in attack phase. The distribution shifts are found in all 6 attacks, but numbers of variables that shift distribution vary. There is no common variable across all 6 attacks. However, the common groups of

variables that shift among the three phases are Process, Processor, Terminal Services Session, and Memory groups.

Table 35 shows an example result of an uncommon variable from EZPublish that shift distribution among three phases. The shift does not happen in other attacks in this study.

Table 35. Example result of uncommon variables from EZPublish.

Variable Name	Pre-attack	Attack	Post-attack
Memory\Cache Faults/sec	Unidentified	Normal	Unidentified

From Table 35, variable "Memory\Cache Faults/sec" does not fall into either uniform, exponential or normal distribution during pre-attack and post-attack phase, but falls into normal distribution during the attack. The variable shows the number of faults which occur when a page sought in the file system cache is not found there and must be retrieved from elsewhere in the memory or from disk. The file system cache is an area of physical memory that stores recently used pages of data for applications. We believe the reason for this distribution shift is that when the attack happens, the attacker requests a confidential file, which is not frequently accessed, from the victim machine. Due to infrequent access, the file is not cached and the system has to fetch it. This increases the faults and also lowers the variance of faults/sec during the attack, and makes the variable fall into normal distribution. The reason that this variable does not show the shift in other attacks like Netbus Trojan or IRC Chat is that these two attacks do not involve requesting the infrequently accessed file.

4.5.1.3 Six Attacks: Correlation of Variables

For correlation of variables, we consider autocorrelation and Pearson correlation. We outline a sampling of our results in this section.

Autocorrelation

"Autocorrelation is the expected value of the product of a random variable or signal realization with a time-shifted version of itself" (http://cnx.rice.edu/content/m10676/latest/). We use autocorrelation analysis to detect whether a variable changes its autocorrelation (shifts) between pre-attack, attack and post-attack phases.

As shown in Figure 22, we discover the variables, which shift autocorrelation between phases, using the ARP Poison attack as an example. From Figure 22 we can see that we collect variables from 3 computers (Attacker, Victim and Bystander), during 3 phases of an attack (pre, during and post) using 2 collection methods (local and remote). We look at whether the autocorrelation for each variable in each phase is high or low, and record those variables that change autocorrelation between phases. The procedure of discovering the shifting variables for other attacks follows a similar routine. The procedure of finding common shifting variables among all six attacks is shown in Figure 23, where we see that we consider all sets of shifting variables for each attack and each phase, and extract those variables that are common (in the union of the sets) for each machine.

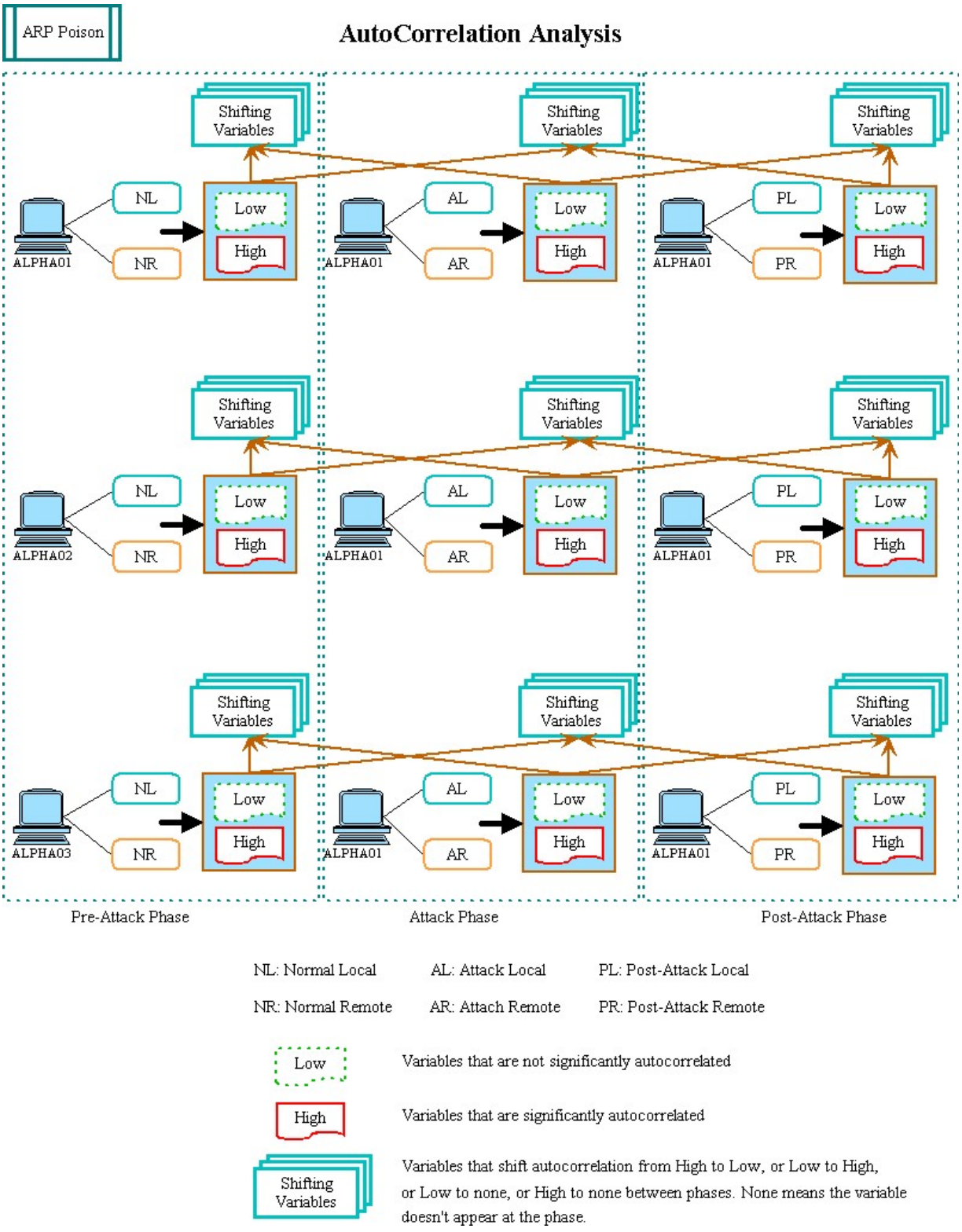


Figure 22. Procedure of finding shifting variables in autocorrelation analysis

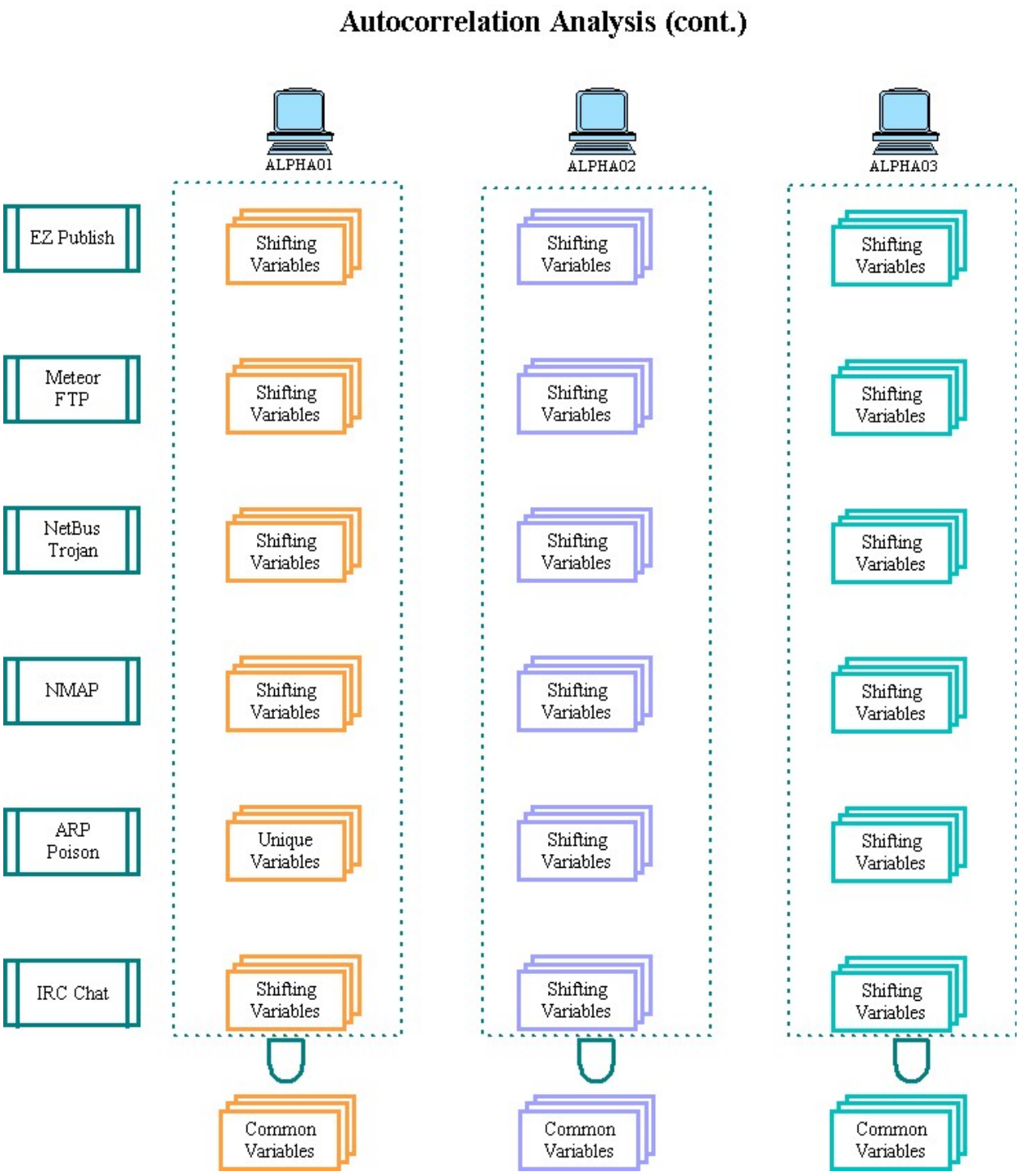


Figure 23. Procedure of finding common variables in autocorrelation analysis

The common variables that shift autocorrelation status among six attacks are shown in Table 36. “H” means the variable is significantly autocorrelated (using tests for correlation with

a p-value of .05 and 10 lags; both commonly used values). “L” means the variable is not significantly autocorrelated. “-” indicates the variable shows that variable is neither “H” nor “L”.

Table 36. Autocorrelation Shifting Variables on machine Victim

VARIABLE	PRE-ATTACK	ATTACK	POST-ATTACK
Terminal Services Session(Console)\Input Errors	H	H	-
Terminal Services Session(Console)\Input Async Overflow	H	H	-
Terminal Services Session(Console)\Total Errors	H	H	-
Terminal Services Session(Console)\Total Async Overflow	H	H	-
Terminal Services Session(Console)\Protocol Bitmap Cache Reads	H	H	-

Table 37 gives an example of an uncommon shifting variable from the ARP Poison attack.

Table 37. Example of an uncommon variable from ARP Poison

VARIABLE	PRE-ATTACK	ATTACK	POST-ATTACK
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec	L	H	L

In Table 37, the variable “Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec” on the victim machine under ARP Poison attack, is not autocorrelated in the pre-attack or post-attack phase. It is highly autocorrelated in the attack phase. This variable shows the rate at which bytes are sent on the interface, including framing characters. In the ARP Poison attack, the attacker sends ARP response packets with the wrong MAC address to the victim, who receives the requests and updates its ARP table. Thus, the victim cannot reach its destination successfully. The data it sends may explain this high correlation characteristic. Other attacks, like EZPublish, don’t involve multiple network packets, which may be why it doesn’t show up in the common variables of all six attacks.

Table 38 shows an example of a common variable among attacks.

Table 38. Example of a common variable

VARIABLE	PRE-ATTACK	ATTACK	POST-ATTACK
Terminal Services Session(Console)\Protocol Bitmap Cache Reads	H	H	-

“Protocol Bitmap Cache Reads show the number of references to the protocol bitmap cache” [35]. This variable shows significant autocorrelation in both pre-attack and attack phases, but not in the post-attack phase. This may reveal that all the attacks influence the number of references to the protocol bitmap cache, which may explain why it changes its autocorrelation pattern for all the attacks.

Pearson Correlation

We use Pearson correlation to analyze the correlation between two variables. The shifting of the Pearson correlation of a pair of variables between phases can help to detect the shift of the phase changes. Figure 24 shows the procedure of finding Pearson correlation shifting variables in the ARP Poison attack. Figure 25 depicts the procedure of finding common variables that shift Pearson correlation status between phases among all six attacks.

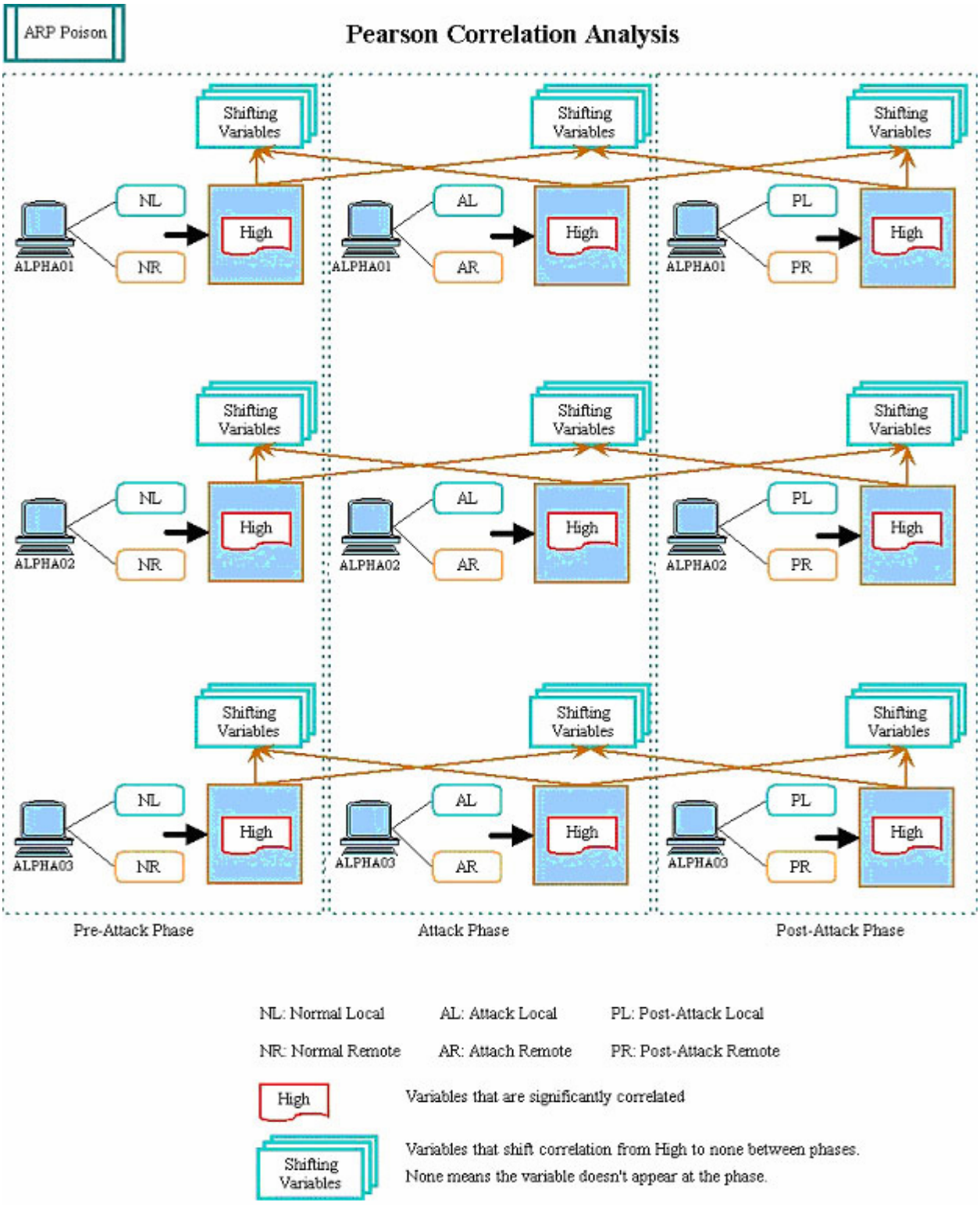


Figure 24. Procedure of finding shifting variable in Pearson correlation analysis

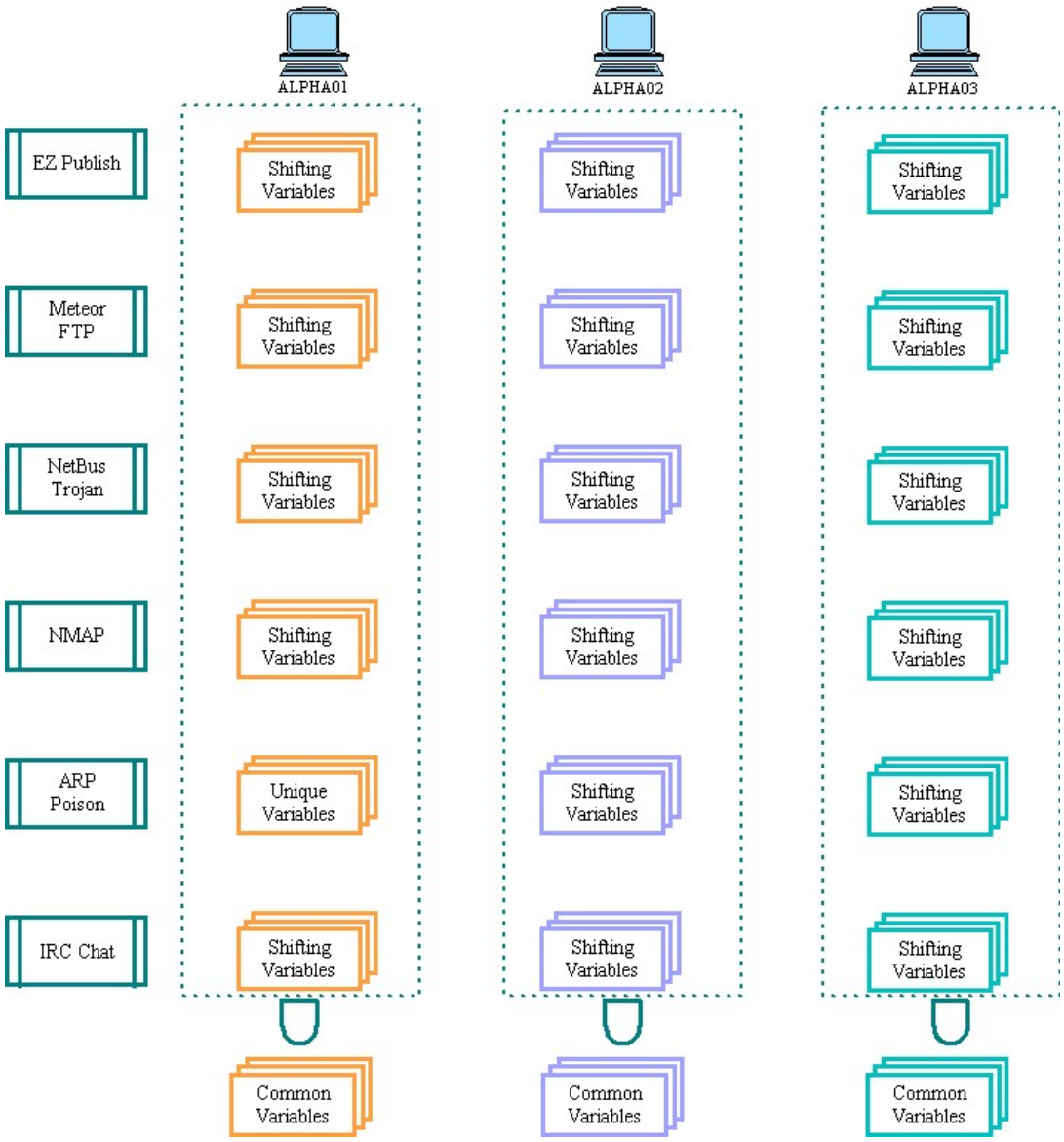


Figure 25. Procedure of finding common shifting variable in Pearson correlation analysis

Compared to Autocorrelation analysis in which we find only 5 common variables, there are 266 pairs of common variables in the six attacks. This is because Pearson correlation analysis looks into pairs of the variables instead of a single variable. For instance, given 500 variables, autocorrelation analysis will analyze the 500 variables while Pearson correlation will analyze $C_{500}^{2} = 124{,}750$ pairs of variables. Table 39 shows the first 10 pairs of common variables. "H" shows the two variables are significantly correlated. "-" indicates the two variables are not.

Table 39. Sample Pearson Correlation Common Variables on Machine Victim

VARIABLE1	VARIABLE2	PRE-ATTACK	ATTACK	POST-ATTACK
Cache\Lazy Write Flushes/sec	Process(SVCHOST#1)\Handle Count	H	-	-
Cache\Lazy Write Pages/sec	Process(SVCHOST#1)\Handle Count	H	-	-
Memory\Page Faults/sec	Objects\Threads	H	-	-
Memory\Page Faults/sec	Process(SVCHOST#1)\Thread Count	H	-	-
Memory\Page Faults/sec	Process(SVCHOST#1)\Pool Paged Bytes	H	-	-
Memory\Page Faults/sec	Process(_Total)\Thread Count	H	-	-
Memory\Page Faults/sec	System\Threads	H	-	-
Memory\Page Faults/sec	Terminal Services Session(Console)\Thread Count	H	-	-
Memory\Available Bytes	Terminal Services Session(Console)\Input Errors	H	-	-
Memory\Available Bytes	Terminal Services Session(Console)\Total Errors	H	-	-

Table 40 gives an example pair of uncommon shifting variables.

Table 40. Example pair of uncommon variables

VARIABLE1	VARIABLE2	PRE-ATTACK	ATTACK	POST-ATTACK
Memory\Page Faults/sec	Redirector\Packets Received/sec	-	H	-

Variables “Memory\Page Faults/sec” and “Redirector\Packets Received/sec” are not correlated in pre or post-attack phases, but are in attack phase.

Memory\Page Faults/sec is the overall rate of page faults handled by the processor per second. A page fault occurs when a process requires code or data that is not in its working set (its space in physical memory). This counter includes both hard faults (those that require disk access) and soft faults (where the faulted page is found elsewhere in physical memory) [35].

The Redirector performance object consists of counters that monitor network connections originating at the local computer. Packets Received/sec is the rate at which the Redirector is receiving packets (also called SMBs or Server Message Blocks). Network transmissions are divided into packets. The average number of bytes received in a packet can be obtained by dividing Bytes Received/sec by this counter [35].

Table 41 gives an example pair of common shifting variables.

Table 41. Example pair of common variables

VARIABLE1	VARIABLE2	PRE-ATTACK	ATTACK	POST-ATTACK
Memory\Page Faults/sec	System\Threads	H	-	-

System\Threads is the number of threads in the computer at the time of data collection. Notice that this is an instantaneous count, not an average over the time interval. A thread is the basic executable entity that can execute instructions in a processor [35].

The above two variables are correlated in the pre-attack phase, which may reveal the "normal" relationship between them. Note that both variables in the pair in Table 41 are also correlated in the pre-attack phase.

4.5.1.4 Six Attacks: Variable Difference in Means

In this section we investigate the differences in average of all variables among the three phases: pre-attack, attack, and post-attack. In earlier versions of attack simulations, we have both active scenarios (with user activity) and inactive scenarios (without user activity). We find that numbers of variables that have a shift in average between phases are noticeably higher in user activity scenarios. These results suggest that if we know characteristics of both signal and noise, it will enable us to see more contrast between the two and thus, detect the intrusion more effectively.

Since our data has several types of distributions, we used the Mann-Whitney U test (Wilcoxon test) to test for difference in means because of its reliable performance regardless the data distributions. In each attack, the Mann-Whitney U test is conducted in three two-phase files: pre-attack vs. attack, attack vs. post-attack, and pre-attack vs. post-attack. Table 42 shows example result from the Mann-Whitney test from UDP Storm attack.

Table 42. Mann-Whitney test from UDP Storm attack

Active /Inactive	Number of significant variables
Active	319
Inactive	189

Table 43 shows example results from the Mann-Whitney test on data from the ARP Poison attack.

Table 43. Example results from Mann-Whitney on ARP Poison attack

Active /Inactive	Number of significant variables
Active	139
Inactive	103

From Table 42 and Table 43, the numbers of significant variables are noticeably higher in active scenarios than in inactive scenarios. In newer version of attack simulation, we have a list of common variables that have mean shift among phases in all 6 attacks Table 44 shows a subset of these variables.

Table 44. Example list of common variables that shift averages

Variable Name
Memory\Available Bytes
Memory\Committed Bytes
Memory\Demand Zero Faults/sec
Memory\Pool Paged Bytes
Memory\Pool Paged Resident Bytes
Memory\% Committed Bytes In Use
Memory\Available Kbytes
Objects\Threads
Objects\Events
Process(CSRSS)\Handle Count
Process(LSASS)\Working Set
Process(SVCHOST#1)\Virtual Bytes
Process(SVCHOST#1)\Working Set
Process(SVCHOST#1)\Page File Bytes
Process(SVCHOST#1)\Private Bytes
Process(SVCHOST#1)\Thread Count
Process(_Total)\Virtual Bytes
Process(_Total)\Working Set Peak
Process(_Total)\Working Set
Process(_Total)\Page File Bytes Peak
Process(_Total)\Page File Bytes
Process(_Total)\Private Bytes
Process(_Total)\Thread Count
Process(_Total)\Pool Nonpaged Bytes
Terminal Services Session(Console)\Virtual Bytes
Terminal Services Session(Console)\Working Set Peak
Terminal Services Session(Console)\Working Set
Terminal Services Session(Console)\Page File Bytes Peak
Terminal Services Session(Console)\Page File Bytes
Terminal Services Session(Console)\Private Bytes
Terminal Services Session(Console)\Thread Count
Terminal Services Session(Console)\Pool Nonpaged Bytes

From Table 44, the common groups of variables that have a mean shift between phases are Memory, Objects, Process, and Terminal Services Session. Most of the common variables are good indicators for the level of activity generated from attacks.

In IRC Chat, the “Objects\Events” variable average value during the attack phase is more than that of pre-attack phase. This variable shows the number of events in the computer at the time of data collection. An event is used when two or more threads wish to synchronize execution. We believe this mean shift happens because, during the attack, the computer opens a new network connection with another machine. So, perhaps, the system creates more threads to handle this network activity. We found this increase in averages in other attacks (i.e. NMAP).

We find many variables that shift average values among phases in only some types of attacks. Table 45 shows an example list of uncommon variables that have changes in average value among phases from NMAP attack.

Table 45. Example list of uncommon variables from NMAP attack.

Variable Name
IP\Datagrams/sec
IP\Datagrams Received/sec
IP\Datagrams Received Delivered/sec
IP\Datagrams Sent/sec
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Total/sec
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets/sec
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Received/sec
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets Received Unicast/sec
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets Sent Unicast/sec
Processor(0)\Interrupts/sec
Processor(0)\DPCs Queued/sec
Processor(0)\% Idle Time
Processor(0)\% C3 Time
Processor(0)\C3 Transitions/sec
TCP\Segments/sec
TCP\Connections Passive
TCP\Connections Reset
TCP\Segments Received/sec
TCP\Segments Sent/sec

From Table 45, in NMAP, the “IP\Datagrams/sec” variable’s average value during pre-attack and post-attack is lower than that of the attack phase. “IP\Datagrams/sec” shows the rate at which IP datagrams are received from or sent to the interfaces, including those in error. We believe that the difference in average is caused by the IP inquiries and responses between

attacker and victim during the port scan. Other attack types in this study do not require such IP packets transferring, so this variable appears significant in NMAP only.

4.5.1.5 Sobig Worm Data Analysis

Because the worm analysis only includes one worm, we cannot do comparisons across worms. We include these results as an example to show how our data analysis tools extend beyond analyzing attacks to the analysis of data collected for worms as well. In this section we present a sample of some of our early findings on analyzing probability distribution and variable correlation in worm data.

Probability distribution of variables

Just as with the six attacks described previously, we conduct the KS and Chi Squared tests on the data from the Sobig worm to test whether the variables fall into any of the three distributions: uniform, exponential, and normal. (Note: data from the attack phase of the remote data collection scenario is not available at this time.) These results are shown in Table 46.

Table 46. Results from KS and Chi-Squared tests on worm

Distributions	KS			Chi-Squared			Total
	Uniform	Exponential	Normal	Uniform	Exponential	Normal	
Pre-attack Local	126	0	5	126	0	0	586
Pre-attack Remote	310	0	0	309	0	0	841
Attack Local	28	0	0	37	0	0	734
Post-attack Local	62	0	27	62	0	0	657
Post-attack Remote	257	0	30	257	0	0	846

Similar to the EZPublish attack data, few or no variables fall into the normal distribution during the pre-attack phase. However, the KS test shows a number of variables that fall into the normal distribution during the post-attack phase (as opposed to during the attack phase in the EZPublish attack).

Correlation of Variables

For the correlation of variables, we again perform autocorrelation and Pearson correlation as before. Because there are so many variables in this study, the results given in this section only include the counts of those variables.

Autocorrelation

Our findings suggest that a number of variables shift autocorrelation status between phases. Furthermore, the number of variables increases with remote data collection. These results are given in Table 47.

Table 47. Autocorrelation results for worm

Collection Location	Phase	Variable Count
Local	Pre	585
Local	Attack	733
Local	Post	656
Remote	Pre	840
Remote	Attack	910
Remote	Post	845

Pearson Correlation

In our findings, it seems like literally half of the variables are correlated. The results also suggest that many variables shift between phases and collection modes as we see for autocorrelation. We give these variable counts in Table 48.

Table 48. Pearson correlation results for worm

	Significant Correlations	Total Cells	Correlation Percentage
Pre-attack Local	76940	170236	45.196%
Pre-attack Remote	202078	351541	57.483%
Attack Local	110146	267546	41.169%
Attack Remote	214872	412686	52.067%
Post-attack Local	92770	214185	43.313%
Post-attack Remote	195904	355746	55.069%

4.5.2 Wavelets

In this section we give some results from our analysis using wavelets. We follow these steps:

1. Identify specific data to collect corresponding to the specified raw data derived from attack profiles
2. Extracting data in the log files collected from each attack using specific programming tools and statistical packages
3. Analyzing the extracted variables

The section proceeds as follows: describe procedures used, present lists of identified and extracted variables (steps 1 and 2), present results of Wavelet and ANOVA analysis (step 3).

4.5.2.1 Procedures

We generalize variables from our attack profiles. We use this list of variables as input to programs, which we have written to extract the variables from our simulation data files.

To test our methods of verifying additional features on data, we analyze all variables and report examples of those results in this section. Two kinds of analysis are carried out: Wavelet analysis and ANOVA for the wavelet coefficients. For both the analysis, remote and local scenarios of data collection are considered .Data from performance log on Victim is collected and only the non-zero, non-invaried set of variables are considered for analysis. In all the attacks the number of variables in this category ranges from 400 to 700. Types of wavelet transforms considered for both the analysis

- Haar transform – useful in approximating to step change pattern.
- Morlet transform – useful in approximating to sine and cosine wave patterns.
- Derivative of Gaussian (DOG)/Mexican hat- useful in approximating to Gaussian noise pattern which is dominant in physical space.
- Paul transform –useful in approximating to narrow changes in sine and cosine and forms a bridge between morlet and derivative of Gaussian which will help in estimating the direction in which pattern changes.

4.5.2.2 Identification and Extraction of Variables

Table 49 gives a list of variables that we have identified from our attack profile generalizations and extracted using tools we created.

Table 49. Variables identified and extracted

Data Type	Identified Variables	Extracted Variables
single source		
1	Raw data: header fields and some data fields (e.g., file name) of each packet Computed variable: total similarity score from comparisons of all fields between consecutive packets	IP packet header -SRC,DEST TCP packet header – SRC_PORT,DEST_PORT (from the network data log, extracted using c program)
2	Raw data: a string indicating the start or termination of a host or network application in the Windows security/system/application log	"Registry, security logs, the start time of each new application" EX: Registry log –Explorer.exe Security log –Image file name like C:\WINDOWS\SYSTEM32\DEFRAG.EXE (extracted using c program)
multiple source, subject to Haar and Complex wavelet analysis		
Intensity measures		

1	Raw data: a variable measuring incoming and outgoing traffic volume per second	1. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets/sec. 2. Network Interface(MS TCP Loopback interface)\Packets/sec 3. Network Interface(Intel[R] PRO_Wireless LAN 2100 3A Mini PCI Adapter - Packet Scheduler Miniport)\Packets/sec" (from performance log)
2	Raw data: packets or variables for traffic volumes per second Computed variable: intensity ratio of incoming traffic volume to outgoing traffic volume per second	1. Network Interface(Intel[R] PRO_Wireless LAN 2100 3A Mini PCI Adapter - Packet Scheduler Miniport)\Packets Received/sec 2. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets Received/sec 3. Network Interface(MS TCP Loopback interface)\Packets Received/sec 4. Network Interface(Intel[R] PRO_Wireless LAN 2100 3A Mini PCI Adapter - Packet Scheduler Miniport)\Packets Sent/sec 5. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets Sent/sec 6. Network Interface(MS TCP Loopback interface)\Packets Sent/sec" (from performance log)
3	Raw data: a sample per second from a variable measuring *network* resource utilization in *CPU*	Not available.
4	Raw data: a sample per second from a variable measuring *network* resource utilization in *CPU by a particular application* such as web server	Not available directly. 'Server bandwidth' for FTP server is the closest available. Similarly, we could have a variable for web service.
5	Raw data: a sample per second from a variable measuring *network* resource utilization in *storage or length of buffer* (directly linked to response time)	Not available
6	Raw data: a sample per second from a variable measuring *network* resource utilization in *storage or length of buffer* (directly linked to response time) *by a particular application* such as web server	Not available

7	Raw data: a sample per second from a variable measuring *network* resource utilization in *communication bandwidth*	Not available (Network interface/ current bandwidth gives the total bandwidth of the interface, not the used bandwidth. Thus, it is not useful)
8	Raw data: a sample per second from a variable measuring *network* resource utilization in *communication bandwidth by a particular application* such as web server	Not available
9	Raw data: "Destination Port" in the TCP header of each *incoming* packet to a host Computed variable: Access intensity (access count per second) to *more common* ports (www, email, etc.)	TCP packet header – SRC_PORT (in the network data log, extracted using c program)
10	Raw data: "Destination Port" in the TCP header of each *outgoing* packet from a host Computed variable: Access intensity (access count per second) to *more common* ports (www, email, etc.)	TCP packet header – DEST_PORT (in the network data log, extracted using c program)
11	Raw data: "Destination Port" in the TCP header of each *incoming* packet to a host Computed variable: Access intensity (access count per second) to *less common* ports (all others)	TCP packet header – SRC_PORT (in the network data log, extracted using c program)
12	Raw data: "Destination Port" in the TCP header of each *outgoing* packet from a host Computed variable: Access intensity (access count per second) to *less common* ports (www, email, others?)	TCP packet header – DEST_PORT (in the network data log, extracted using c program)
13	Raw data: "Event Type" of each audit event record on a host Computed variable: Intensity (number of events per second) of *more common* event types	In each of application/system/security logs – the "event id field number (extracted using program)
14	Raw data: "Event Type" of each audit event record on a host Computed variable: Intensity (number of events per second) of *less common* event types (all others)	In each of application/system/security logs – the "event id field number (extracted using program)
15	Raw data: a variable measuring *host* resource utilization (a sample per second) in *CPU*	"Processor(_Total)\% Processor Time" - from performance log

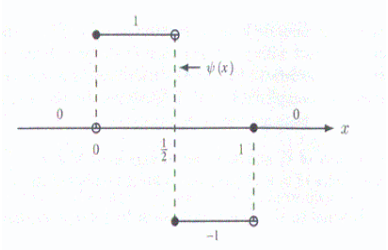
16	Raw data: a variable measuring *host* resource utilization (a sample per second) in *storage*	Not available
Activity pattern measures		
17	Raw data: "Destination Port" in the TCP header of each *incoming* packet to a host Computed variable: Frequency ratio of more common ports (web, email, and others?) to less common ports (all others) on a host Computation method: 1) initialize the ratio with the average ratio in a noise condition, 2) update the ratio with each packet using "Destination Port" and $EWMA_n$=(1 if common or 0 if not + 0.3*$EWMA_{n-1}$)/ (1 if uncommon or 0 if not + 0.3*$EWMA_{n-1}$)	TCP packet header – SRC_PORT" (in the network data log, extracted using c program)
18	Raw data: "Destination Port" in the TCP header of each *outgoing* packet from a host Computed variable: Frequency ratio of more common ports (define?) to less common ports (all others) on a host Computation method: 1) initialize the ratio with the average ratio in a noise condition, 2) update the ratio with each packet using "Destination Port" and $EWMA_n$=(1 if common or 0 if not + 0.3*$EWMA_{n-1}$)/ (1 if uncommon or 0 if not + 0.3*$EWMA_{n-1}$)	TCP packet header – DEST_PORT (in the network data log, extracted using c program)
19	Raw data: "Event Type" of each audit event record on a host Computed variable: Frequency ratio of more common event types to less common event types (all others) on a host Computation method: 1) initialize the ratio with the average ratio in a noise condition, 2) update the ratio with each event using "Event Type" and $EWMA_n$=(1 if common or 0 if not + 0.3*$EWMA_{n-1}$)/ (1 if uncommon or 0 if not + 0.3*$EWMA_{n-1}$)	"Event id field number" extracted from security, system and application log. (extracted using c program and also the computational method defined has been implemented)

20	Raw data: Each string entry in the Windows Security/Application/System log Computed variable: Frequency ratio of more common entry types to less common entry types (all others) on a host Computation method: 1) initialize the ratio with the average ratio in a noise condition, 2) update the ratio with each entry using the entry type and $EWMA_n$=(1 if common or 0 if not + 0.3*$EWMA_{n-1}$)/ (1 if uncommon or 0 if not + 0.3*$EWMA_{n-1}$)	Registry, security logs, the start of each new string EX: Registry log –Explorer.exe Security log –Image file name like C:\WINDOWS\SYSTEM32\DEFRAG.EXE
21	Raw data: each file name appearing in ? log or counter on a host Computed variable: Frequency ratio of common files to less common files (all others) on a host Computation method: 1) initialize the ratio with the average ratio in a noise condition, 2) update the ratio with each file name appearance using "Event Type" and $EWMA_n$=(1 if common or 0 if not + 0.3*$EWMA_{n-1}$)/ (1 if uncommon or 0 if not + 0.3*$EWMA_{n-1}$)	Security log –Image file name like C:\WINDOWS\SYSTEM32\DEFRAG.EXE
22	Raw data: "Event Type" of each audit event record on a host Computed variable: Frequency ratio of more common event types to less common event types (all others) on a host Computation method: 1) initialize the ratio with the average ratio in a noise condition, 2) update the ratio with each file name appearamce using "Event Type" and $EWMA_n$=(1 if common or 0 if not + 0.3*$EWMA_{n-1}$)/ (1 if uncommon or 0 if not + 0.3*$EWMA_{n-1}$)	Done already in 19

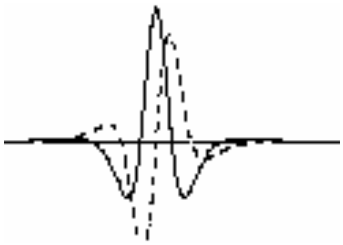
4.5.2.3 Wavelet Analysis

The purpose of wavelet analysis is to convert the time series data into frequency (retaining the time domain information) and analyze the pattern of the input data and also the different frequency components and the signal strengths of them.

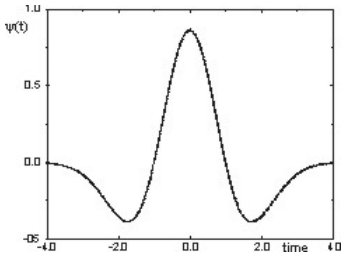
The wavelet shapes of the transformations we use are shown in Figure 26.



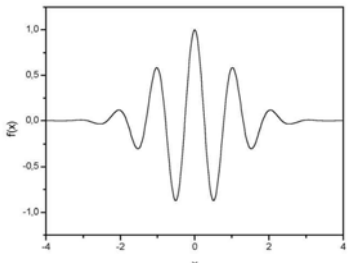
a) Haar



b) Paul



c) DOG



d) Morlet

Figure 26. Wavelet shapes of transformations

We describe the method adapted for implementing each wavelet transform next.

Haar: The transformation into the additive and difference components was done successively till the nearest power of 2 in the observations for each variable in the performance log and the frequency components were defined such that the difference terms in each iteration forms one set of frequency.

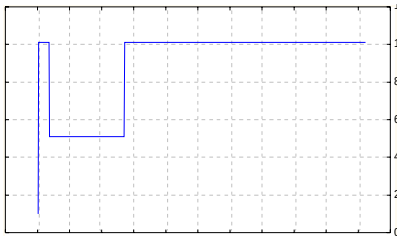
Morlet,Paul and DOG: The transformation was done as per the equation of the mother wavelet for each of the 3 transforms. Totally 29 different scales (each representing one frequency component) were designed ranging from 2 to 256 to convert time series data into frequency.

We draw two types of results for this analysis: Pattern based on visual inspection of time series plot and variations in signal strength based on visual inspection of wavelet power spectrum.

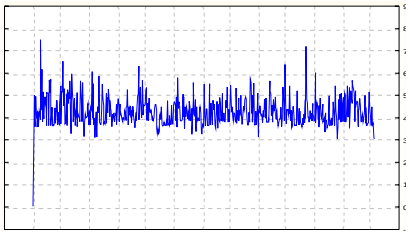
For pattern based on visual inspection of time series plot, the results indicate the following patterns in the numerical data obtained from the performance log across all the 6 attacks considered.

- Step change 10-15%
- Random fluctuations 25-35%
- Spike change 40-45%
- Steady change 2 to 5%

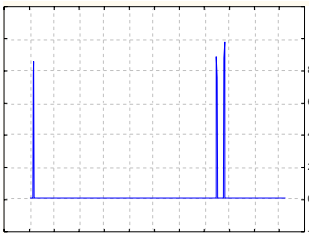
For these overall percentages, the pattern looks the same in all 3 phases, where as individual variables may differ between each phase. These patterns are shown in Figure 27, which just gives an example of the basic shape of the pattern, and is not meant to be considered in detail.



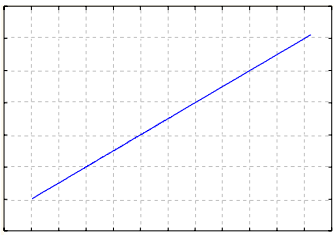
a) Step change



b) Random fluctuations



c) Spike



d) Steady change

Figure 27. Examples of basic shapes of signal patterns

For variations in signal strength based on visual inspection of wavelet power spectrum, the signal strength is analyzed based on how frequently it changes and in which frequency band it falls.

The tables below will indicate the presence of variations in signal strength in 3 different frequency bands.

1. High pass band –indicating that the variations in the time series data are rapid and one has to look at high frequency components to analyze the data. .
2. Medium pass band –indicating that the fluctuations in the time series data are moderate and one has to look at medium frequency components to analyze the data.
3. Low pass band - indicating that the fluctuations in the time series data are slow and one has to look low frequency components to analyze the data.

Note: Low frequency components can be used as a characteristic to detect slow and stealthy versions of a cyber attack.

For the observations in the table, uniformly spread in all bands indicates that time series data has all frequency components with equal strengths and one has to analyze all of them. Dominant high/low pass band indicates that these particular frequency components are strong in the time series data and the data can be analyzed with these components. Low and high pass band indicates that both low and high frequency components are strong and the data can be analyzed with them.

Table 50-Table 55 give the analysis results for pattern and signal strength covering all wavelet methods and variables.

Table 50. EZPublish Attack

PHASE	PRE ATTACK	ATTACK	POST ATTACK
Pattern based on visual inspection of time series plot	Spike, step change, random fluctuations and steady change	Spike, step change, random fluctuations and steady change	Spike, step change, random fluctuations and steady change
Variations in signal strength based on visual inspection of wavelet power spectrum	Uniformly spread in all bands	Low and high pass bands	Uniformly spread in all bands

Table 51. NMAP Scanner Attack

PHASE	PRE ATTACK	ATTACK	POST ATTACK
Pattern based on visual inspection of time series plot	Spike, step change and random fluctuations. One variable shows steady increase.	Spike, step change, random fluctuations and steady change	Spike, step change and random fluctuations and steady increase
Variations in signal strength based on visual inspection of wavelet power spectrum	Uniformly spread in all bands	Dominant high pass band	Uniformly spread in all bands

Table 52 Netbus Trojan Attack

PHASE	PRE ATTACK	ATTACK	POST ATTACK
Pattern based on visual inspection of time series plot	Spike, step change, random fluctuations and steady increase	Spike, step change, random fluctuations and steady change	Spike, step change, random fluctuations and steady increase
Variations in signal strength based on visual inspection of wavelet power spectrum	Uniformly spread in all bands	Low and high pass band	Uniformly spread in all bands

Table 53. Meteor FTP Attack

PHASE	PRE ATTACK	ATTACK	POST ATTACK
Pattern based on visual inspection of time series plot	Spike, step change , random fluctuations and steady increase	Spike, step change, random fluctuations and steady change	Spike, step change , random fluctuations and steady increase
Variations in signal strength based on visual inspection of wavelet power spectrum	Uniformly spread in all bands	Dominant high pass band	Uniformly spread in all bands

Table 54. IRC Chat Attack

PHASE	PRE ATTACK	ATTACK	POST ATTACK
Pattern based on visual inspection of time series plot	Spike, step change , random fluctuations and steady increase	Spike, step change, random fluctuations and steady change	Spike, step change , random fluctuations and steady increase
Variations in signal strength based on visual inspection of wavelet power spectrum	Uniformly spread in all bands	Dominant low pass band	Uniformly spread in all bands

Table 55. ARP Poison Attack

PHASE	PRE ATTACK	ATTACK	POST ATTACK
Pattern based on visual inspection of time series plot	Spike, step change , random fluctuations and steady increase	Spike, step change, random fluctuations and steady change	Spike, step change , random fluctuations and steady increase
Variations in signal strength based on visual inspection of wavelet power spectrum	Uniformly spread in all bands	Dominant low pass band	Uniformly spread in all bands

For each attack, we describe an example variable with an observed pattern that changes between phases.

EZPublish: Example variable, physical disk - idle time (step decrease), involves reading a file from the victim machine. Physical disk/idle time represents percentage of time in sample interval that the disk is idle. During normal phase, the only activity on disk is writing the log files. During the attack, a new file is accessed, causing a step down change in this variable.

NMAP Scanner: Example variable, connections reset/sec (staircase). NMAP sends SYN packets to different ports and resets connections once it receives a response. This leads to an increase in connections reset/sec in NMAP. Thus, this is a staircase like step increase.

Netbus Trojan: Example variable, process (svchost) IO write operations per sec (step increase), represents amount of data written to a remote destination over the network. This represents the screen dump data sent to the attacker from the victim, after netbus is installed. Screen dump is just an image file showing the current desktop of the victim to the attacker.

Meteor FTP: No significant change can be noticed in variables in performance log for this attack. The attack only involves one string sent over the network and crash in FTP server. However, since the normal phase does not have any network/ftp server activity, no difference is seen.

IRC Chat: Example variable, network interface packets sent/sec (up-spike at regular intervals). Pings are sent between client/server at regular intervals to maintain connection. This leads to a spike at regular intervals in this variable.

ARP Poison: Since ARP update packets are the only network activity, this is not reflected in performance logs. Thus, no significant change can be seen in any variable.

4.5.2.4 ANOVA Analysis

In general, the purpose of analysis of variance (ANOVA) is to test for significant differences between averages. If we are only comparing two averages, then ANOVA will give the same results as the t-test for independent samples (if we are comparing two different groups of cases or observations), or the t-test for dependent samples (if we are comparing two variables in one set of cases or observations)

The method we use considers only the main effects, ignoring the interaction term. We analyze the first order (non interactive) effect of 2 categorical independent variables (phase and frequency) on the dependent variable (response) which is the energy of wavelet coefficients.

The independent variables (factors) are

- Phase – pre attack, attack and post attack
- Frequency – 29 different scales (each representing a frequency) of the mother wavelet for 3 types of wavelets Morlet, Derivative of Gaussian and Paul.

The dependent variable is:

- Energy of the wavelet coefficients (sum of the squares) for each of the 29 different frequency components and 3 different phases.

The input to our ANOVA analysis is thus described in Table 56.

Table 56. Input table to ANOVA

Phase	Frequency	Energy value(response)
1 1 . . 1	1 2 . . 29	Values corresponding to each frequency (total 29) for pre attack phase
2 2 . . 2	1 2 . . 29	Values corresponding to each frequency (total 29) for attack phase
3 3 . . 3	1 2 . . 29	Values corresponding to each frequency (total 29) for post attack phase

Table 57-Table 62 give the number of variables found significant in the ANOVA main effects analysis for each attack, using the independent variables described above. Following each table is a summary of the findings. Here, again, we are presenting our findings. A full discussion and critical evaluation of all findings is outside the scope of this project and only included in our future work.

Table 57. EZPublish Attack

Type of wavelet		Morlet			DOG/Mexican hat			Paul		
Machine		01	02	03	01	02	03	01	02	03
Percentage significant variables (Local)	Phase	78	72.8	68.8	81	80.3	80.3	80.8	79.2	74.5
	Frequency	19.1	19.6	20.6	22.2	21	15.9	20.2	22.1	20.1
Percentage significant variables (Remote)	Phase	85.1	58.3	50.4	91.6	61.8	60.1	90.5	62	57.2
	Frequency	19.4	16.7	10.7	13.6	13	7.7	20.4	16.5	8.1

Table 57 summary:

- Significant variables are very high in phase factor compared to frequency.
- 75 to 80% of input variables are significant in phase on an average.
- 10 to 20% of input variables are significant in frequency on an average.
- Compared to Morlet, Paul and DOG wavelets show more significant variables in phase.

Table 58 NMAP Scanner Attack

Type of wavelet		Morlet			DOG/Mexican hat			Paul		
Machine		01	02	03	01	02	03	01	02	03
Percentage of significant variables (Local)	Phase	46	44.4	60.3	62.4	73.1	87.2	58.6	66.9	84.1
	Frequency	30.7	32.7	37.5	26.7	25.5	35.3	32.4	33.6	39.2
Percentage of significant variables (Remote)	Phase	61.1	51.5	60.4	83.5	71.6	89.1	74.3	69.2	79.7
	Frequency	44.8	35.1	33.3	31.7	19.4	22.6	49.8	36	37.7

Table 58 Summary:

- Significant variables are higher in phase factor compared to frequency.
- 65-70% of input variables are significant in phase on an average.
- 25-35% of input variables are significant in frequency on an average.
- There is a large difference in the number of significant variables in DOG wavelet (remote) between phase and frequency compared to other two.

Table 59. Netbus Trojan Attack

Type of wavelet		Morlet			DOG/Mexican hat			Paul		
Machine		01	02	03	01	02	03	01	02	03
Percentage of significant variables (Local)	Phase	68.7	59.4	51.7	90.6	88.1	77.8	84.9	79.8	72.1
	Frequency	39.5	30	31.4	36.1	22.2	23.7	41.6	31.8	32.3
Percentage of significant variables (Remote)	Phase	64.8	69.6	57.5	87.1	88.1	55.3	77.1	77.7	74.4
	Frequency	48.4	35.9	22.4	29.7	23.3	15.9	49.3	37.1	31.2

Table 59 Summary:

- Significant variables are very high in phase factor compared to frequency.
- 70-75% of input variables are significant in phase on an average.
- 25-35% of input variables are significant in frequency on an average.
- Significant variables in frequency are relatively high compared to other attacks.
- DOG shows higher significant variables in phase compared to other two in local and remote scenario.

Table 60. Meteor FTP Attack

Type of wavelet		Morlet			DOG/Mexican hat			Paul		
Machine		01	02	03	01	02	03	01	02	03
Percentage of significant variables (Local)	Phase	59	60.7	69	61.3	68.3	84.2	61.2	61.9	77.2
	Frequency	17	21.3	22.9	15.2	14.9	20.5	18.7	23.7	22.7
Percentage of significant variables (Remote)	Phase	79.8	70.8	81.7	81.9	84.3	89.1	81.7	76.3	85.1
	Frequency	19.9	27.3	33.9	16.5	20.6	27.3	18.9	24.6	35.8

Table 60 Summary:

- Significant variables are very high in phase factor compared to frequency.
- 65-75% of input variables are significant in phase on an average.
- 20-25% of input variables are significant in frequency on an average.
- In DOG the significant variables for frequency factor are less compared to the other two.

Table 61. IRC Chat Attack

Type of wavelet		Morlet			DOG/Mexican hat			Paul		
Machine		01	02	03	01	02	03	01	02	03
Percentage of significant variables (Local)	Phase	70.4	58.7	42.3	94	92.2	80.4	81.3	77.4	68.4
	Frequency	39.9	34.8	31.3	33.4	29.9	29.5	40.6	37.9	34.2
Percentage of significant variables (Remote)	Phase	61.9	67.9	44.2	88.9	92.7	81.9	77.1	83.6	75
	Frequency	37.2	42.4	34.9	31	39.1	32.6	39.5	46.4	40.2

Table 61 Summary:

- Significant variables are high in phase factor compared to frequency.
- 70-80% of input variables are significant in phase on an average.
- 30-40% of input variables are significant in frequency on an average.
- Number of significant variables for frequency factor is higher than all other attacks except ARP poison and Netbus trojan.

Table 62. ARP Poison Attack

Type of wavelet		Morlet			DOG/Mexican hat			Paul		
Machine		01	02	03	01	02	03	01	02	03
Percentage of significant variables (Local)	Phase	47.3	54.5	49.2	69.4	82.9	84.3	61.4	71.2	79.1
	Frequency	25.7	23.6	31.9	24.4	21.9	24.7	25.3	28.3	32.5
Percentage of significant variables (Remote)	Phase	69.1	34.6	52.9	94.3	55.1	75.8	83.8	48.1	65.2
	Frequency	43.4	19.7	22.2	37.5	22.5	17.5	44.3	22.6	24.7

Table 62 Summary:

- Significant variables are high in phase factor compared to frequency
- 60-70% of input variables are significant in phase on an average.
- 25-30% of input variables are significant in frequency on an average.
- Number of variables significant in phase is much higher for DOG compared to other two.

We again give an example variable from each attack that shifts between phases and include a “best guess” at why this is so based on our attack profiling effort.

Example of variables for EZPublish results are for phase, network interface packets/sec and frequency: TCP connections reset. Here the client connects to the server and reads a file back. This involves a TCP connection being established. Hence variables show a difference in the attack phase. In the pre attack phase, there are no connections made so no data transfers.

Example of variables for NMAP results are for phase, TCP connections established and frequency, TCP Connections reset. In Nmap, a series of TCP connections are made and reset during the attack. Thus, the variables TCP connections established (phase) ad connections reset (freq) can be used to differentiate the attack from the other two phases.

Example of variables for Netbus results are for phase, network interface packets sent/sec and frequency, network interface bytes sent/sec. In the Netbus Trojan attack, an installation file is copied from attacker to victim leading to network data transfer. Also, the victim’s screendump is sent to the attacker over the network in the attack phase. Thus, both these variables differ between attack phase and pre attack phase.

Example of variables for Meteor FTP results are for phase, network interface packets sent/sec and received/sec and frequency, network interface packets / sec. These variables reflect network activity. During the attack, the attacker sends a long string over the network, which is reflected in these variables. During the pre attack/post attack phases, there is no such activity.

Example of variables for IRC Chat results are for phase, TCP connections established. In this attack, clients connect to the server during the attack phase. This increases the number of connections established at the server. This variable can thus be used to distinguish attack phase from normal phase. For frequency, network interface packets sent/ sec. During the attack, after

connection is established between clients and server, ping packets are sent at regular intervals, which are reflected in the above variable.

Example of variables for ARP Poison results are for phase, network interface packets sent/ sec. In this attack, an ARP packet is sent at regular intervals to the victim machine. Thus, this reflects in the network interface packets sent/sec variable in the attack phase. In the pre attack and post attack phases, there is no such activity. Thus this variable can be used to differentiate phases. For frequency, process (svchost) IO other bytes /sec. This variable shows the amount of bytes sent over the network through thesvchost processes. This is indicative of the ARP packets received during the attack. These packets are not received during the pre attack/post attack phases.

4.6 Summary

This section gives results of our research on discovering the characteristics of cyber signal and noise for cyber signal detection. We describe how we go from attack profiling (our previous section), to generalizing the DFCs of attacks, to applying knowledge from physical space signal detection to the analysis of data and verification of cyber signals. In our analysis, we have applied new techniques for detecting cyber attack observables, and discovered a number of DFCs that are useful in detecting and identifying these observables. Detecting a single observable is just the basis for developing a suite of cyber sensors. Many observables for cyber attacks are also found in normal data. This is why we need to develop a full understanding of the characteristics of both attack and norm data, and then group sensors (based on these observables) in such a way that false alarms are reduced, while maintaining detection accuracy.

We provide a sampling of our data analysis results. The actual amount of results that we have collected thus far is well beyond the scope of this report. Therefore, we attempt to summarize some key examples of how we are using the data we have collected to extract characteristics of cyber signals. The purpose of this investigation is to find characteristics and/or groups of characteristics that can uniquely identify attacks or classes of attacks. We use these results to build our sensor models, as described in the next section.

5. Sensor and Sensor Fusion Models

In the last section on the discovery of characteristics of cyber signal and noise we described how to go from attack profiling, to generalizing the DFCs of attacks, to applying knowledge from physical space signal detection to the analysis of cyber attack data. That section gives a sample of our results on the analytical discovery of characteristics of cyber signals. Previous sections on attack profiling and the analytical discovery of cyber attack and normal use characteristics provide us with the ability to develop sensor models and sensor fusion models for cyber attack detection. First we give some relevant background material. Then provide examples of sensor and sensor fusion models. Finally, we conclude this section.

5.1. Background

In this section we give the background information necessary to understand the sensor and sensor fusion models presented in this section. We first introduce 2 user activities to our dataset. We then present the 3 previously described attacks we consider for this work. Finally, we outline the analytical discovery methods used here.

5.1.1 Activity Data

To design and test a sensor model, in addition to attack data characteristics, we require the characteristics of normal user activity data. For this section, we consider 2 common activities: web browsing and text editing. During the simulation of an attack as described in previous sections, we have a user conducting the respective activity at the victim machine. The user activity continues throughout the simulation phases. This way, we are able to collect data with only user activity, and data with both attack and user activity. With the user activity data we discover characteristics of normal use (noise). The analysis for user activity is the same as the analysis we reported for attack activity. We use the analytical results from both user activity and attack data to build models. The combined data is for testing our models.

5.1.2 Attack Data

We use data collected from 3 attacks simulated in the lab: ARP Poison, EZPublish Confidentiality and NMAP Scanner. These attacks are simulated without user activity, as stated previously, to discover attack characteristics. We choose these 3 attacks arbitrarily from our attack data set and simulate them again with user activity for testing. (We do not include testing on all 6 attacks in this report due to time constraints).

5.1.3 Analytical Discovery

Previously we described how to profile attacks and obtain the DFC of an attack. We simulated several attacks and, using analytical discovery methods, derived matrices of observed DFCs for each attack. For each DFC, we have multiple choices of detection method. In this section, we consider the DFCs for the 3 attacks described above. We select analysis methods to build our sensor models upon and describe them below. These two methods use the techniques: Paul wavelet, cuscore and autocorrelation. These are described previously in detail and summarized here.

Method 1 – Wavelet with Cuscore
Wavelet analysis converts time series data into frequency and analyzes the pattern of the input data, and the different frequency components and their signal strengths. We choose the Paul wavelet transformation based on its superior performance in our previous analysis.

Method 2 - Autocorrelation

"Autocorrelation is the expected value of the product of a random variable or signal realization with a time-shifted version of itself" (http://cnx.rice.edu/content/m10676/latest/). We use autocorrelation analysis to detect whether a variable changes its autocorrelation between the pre-attack (pre-activity) and attack (activity) phases of our simulation.

5.2 Sensor Models

We developed sensor models using the data we colleted in our attack simulations. The sensors are first developed and tested with offline data, as shown in Figure 28. After finalizing a sensor model offline, we verify that the sensor works online, with data collected in real time during an attack. Our online simulations have a similar setup to offline simulations. The only difference is that the sensor models are reading data as it occurs, and reacting accordingly, instead of waiting until all data is collected.

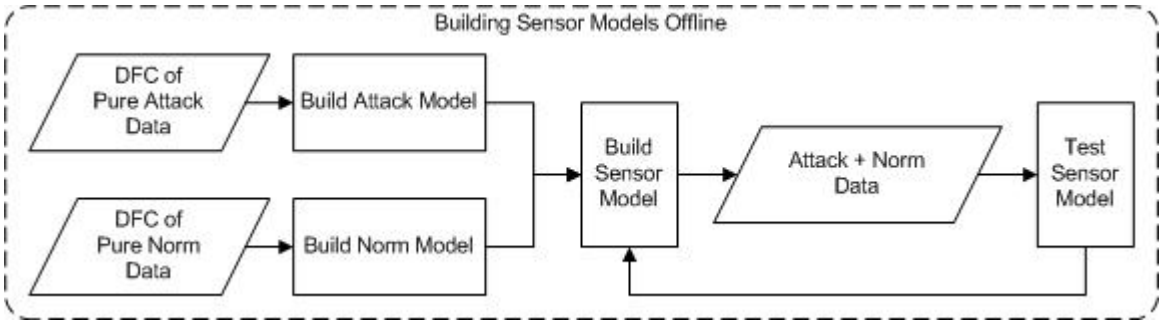


Figure 28. Building sensor models offline

We develop sensor models using data from 3 attacks and the DFC and detection techniques described in the previous section. We include the 2 user activities in our testing phase. We first build a model using Paul wavelets and the Cuscore statistic. We then build a model using autocorrelation and the Cuscore statistic. Table 63 describes the DFC mapping for the sensor models presented in this section.

Table 63. DFC Mapping for Sensor Models

Sensor Model	***Paul Wavelet & Cuscore***	
	Noise Model	**Attack Model**
Data	Raw data variable	Raw data variable
Feature	y = Energy Frequency	y = Energy Frequency
Characteristic	$y_t = T + a_{t0}$	$y_t = T + \delta + a_t$
Sensor Model	***Autocorrelation & Cuscore***	
	Noise Model	**Attack Model**
Data	Raw data variable	Raw data variable
Feature	y = Mean # of significant autocorrelation functions	y = Mean # of significant autocorrelation functions
Characteristic	$y_t = T + a_{t0}$	$y_t = T + \delta + a_t$

5.2.1 Model Based on Paul Wavelet & Cuscore Statistic

The steps outlined below describe our first detection model. These steps are also outlined in Figure 29:

1. From our analysis previously detailed, we find that Paul is the most useful wavelet because most of the data form a spike pattern.
2. We pick variables discovered through sensor optimization (following section) and begin by extracting the variables from our data sets individually and storing them as a text file. This input file is a time series data with 500-600 observations.
3. The input is given to a wavelet program to calculate the Paul wavelet coefficients.
4. Once data is transformed into the wavelet domain, the wavelet coefficients are converted into energies by squaring and summing. These calculated energy values in the wavelet domain define the feature that is tested in this model.
5. The characteristic we observe for all selected variables is a step change. Thus, we choose the Cuscore statistic for step change as our detection model.
6. This model is basically the summation at each point of the difference between each observation and a threshold obtained from the normal scenario. This model has an implicit noise filtering component because the difference is calculated based on a threshold derived from the normal model.
7. The calculated wavelet energy values form the observations for Cuscore. We obtain cuscore values and plot them to identify the step change.

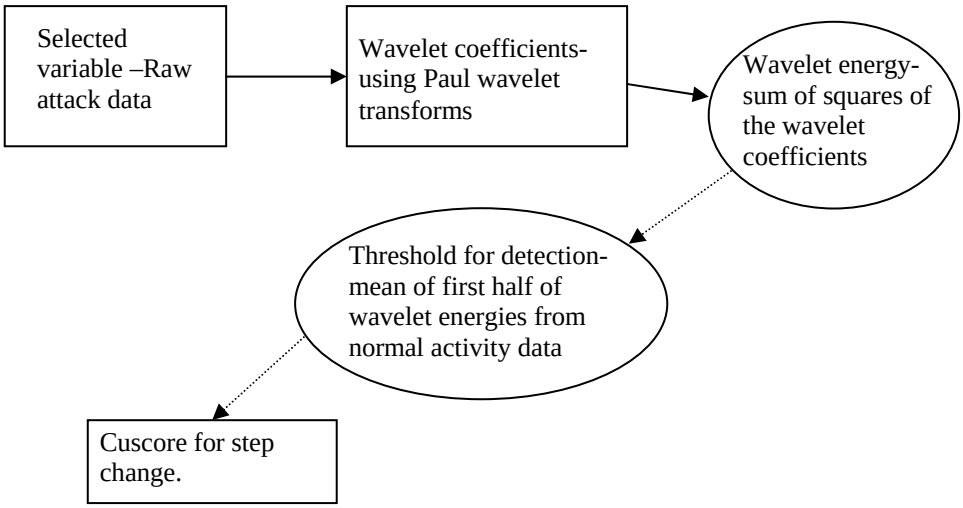


Figure 29. Steps for the Paul wavelet & Cuscore statistic model.

5.2.2 Model Based on Autocorrelation & Cuscore Statistic

Our next detection model uses autocorrelation with the same Cuscore for step change as used in the previous model. The steps are outlined as follows and shown in

Figure 30:

1. We choose variables and analytic discovery methods based on our sensor optimization study (following section).
2. We extract variables and store them in text file formats.
3. We separate the extracted normal activity datasets into two sections. The first half serves as training data to calculate the normal mean in the Cuscore model, while the latter is used, along with data from normal+attack activity, for testing.
4. The numbers of significant autocorrelation functions are calculated on both data using the moving window method, with a window of size 60 observations. The means of the numbers of significant autocorrelation functions are calculated for the input to Cuscore.
5. We plot the Cuscore results for each sensor to observe false alarms, signal indications and first indications of attacks.

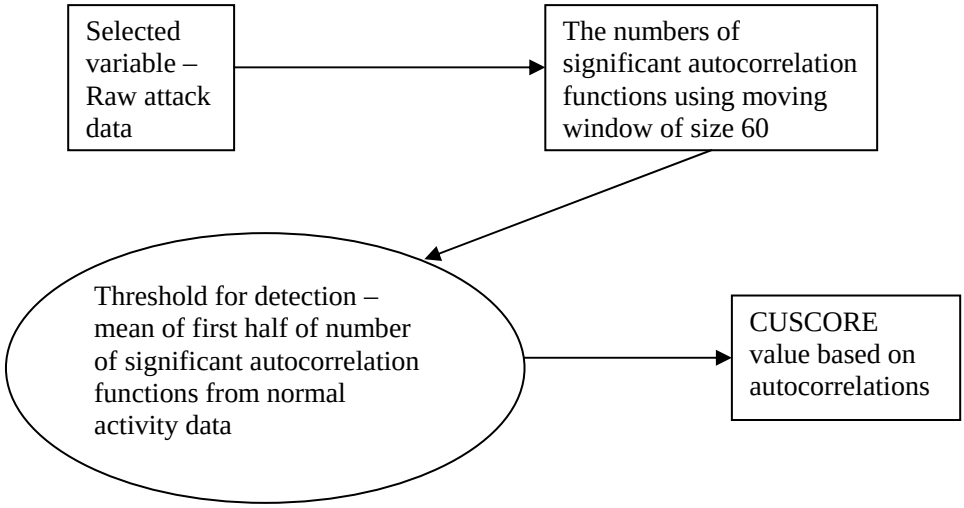


Figure 30. Steps for the autocorrelation & Cuscore statistic model.

5.3 Sensor Fusion Models

We develop our fusion and decision theory based on the unique vector or specific knowledge about the attack rather than a general, existing decision or fusion theory. To perform sensor fusion, we consider the results of our 2 sensor models described previously. During our attack simulations, we collect data both locally at the victim computer, and remotely. For remote data collection, performance data from the victim machine is sent across the network and logged at another location. We present our results for both collection methods for these sensor models. Our threefold objective is to determine which sensor model gives: The highest detection rate, the lowest false alarm rate, and the earliest signal detection

In this section we present the results for testing observables identified in our sensor optimization efforts (following section). These results include specific variables and detection methods identified as sufficient to detect and differentiate the 3 attacks with 2 user activities described previously.

Table 64 lists the sensors we develop and tests completed for each sensor.

Table 64. Sensor testing outline

Sensor #	Sensor Model	Data Variable	Attack	Activity	Data Collection
1	Paul Wavelet	Process(_total)IO other operations/sec	ARP Poison	Web Browsing	Local
					Remote
				Text Editing	Local
					Remote
2	Paul Wavelet	TCP\Segments/sec	EZ Publish	Web Browsing	Local
					Remote
				Text Editing	Local
					Remote
3	Autocorrelation	Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec	ARP Poison	Web Browsing	Local
					Remote
				Text Editing	Local
					Remote
4	Autocorrelation	Terminal Services Session(Console)\Page Faults/sec	EZ Publish	Web Browsing	Local
					Remote
				Text Editing	Local
					Remote
			NMAP	Web Browsing	Local
					Remote
				Text Editing	Local
					Remote

We have checked that all characteristics for the sensors in Table 64 do not appear in the characteristics for normal activities, and thus are indicative of an attack for our data set. In this section, we first present the testing results for our sensor models, and then offer some observations for sensor fusion on these sensors.

5.3.1. Testing Results

This section separates the results obtained by each of our 3 sensor models. Each section includes the Cuscore charts for each variable, attack, activity, and local/remote configuration. For each plot in this section, the following descriptions hold:

1. The x-axis is observations in time.
2. The y-axis is the Cuscore value.
3. The first 287 observations are normal activity data to check false alarm pattern and the rest of the observations form the attack data.
4. The figure name gives the name of the observed variable, the attack and user activity present, and whether the data is from a local or remote collection.

We use the results shown in Figures 30-49 to complete Table 65 and Table 66, from which we make observations for sensor fusion.

5.3.1.1 Paul Wavelet & Cuscore Statistic

The Cuscore results for this model are shown in Figure 31-Figure 38.

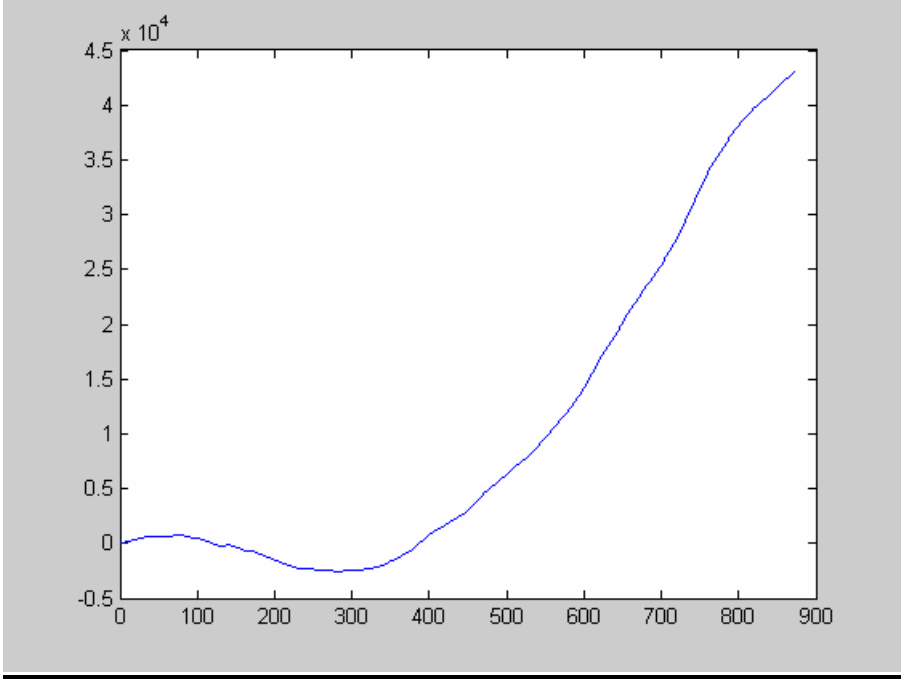


Figure 31. Process(_total)IO other operations/sec, ARP Poison, Web Browsing, Local.

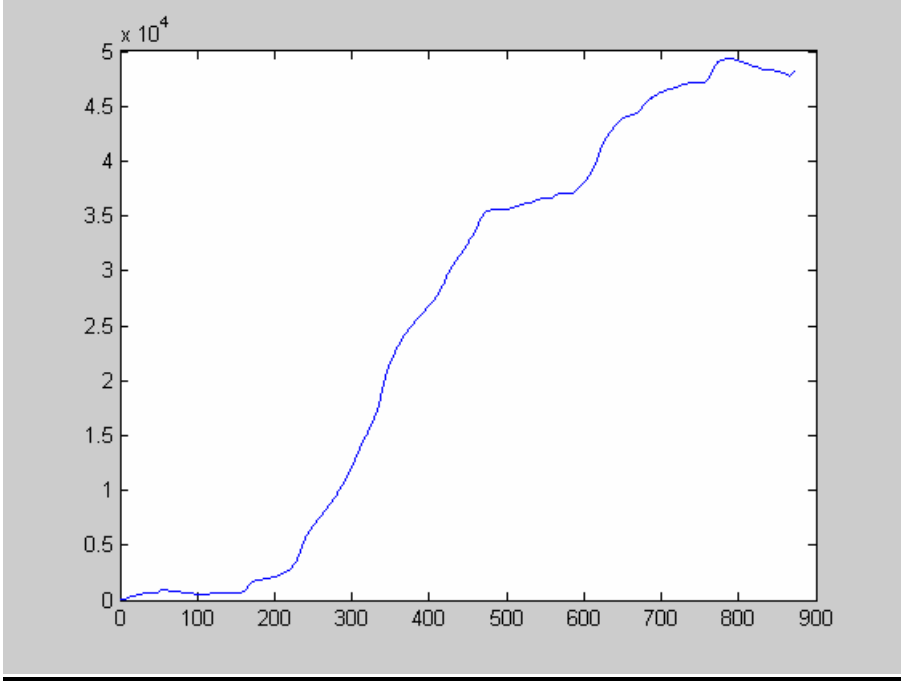


Figure 32. Process(_total)IO other operations/sec, ARP Poison, Web Browsing, Remote.

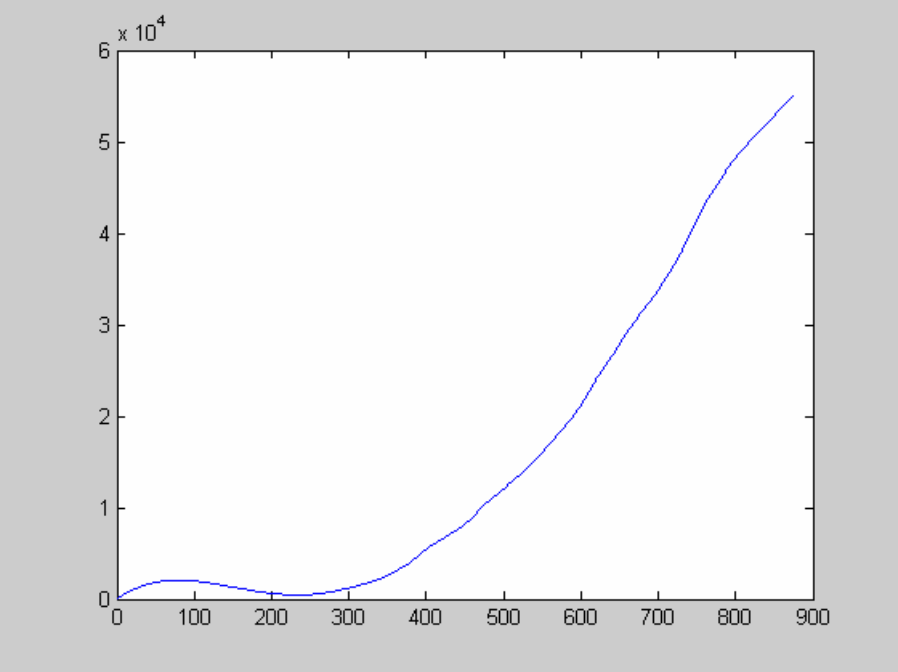


Figure 33. Process(_total)IO other operations/sec, ARP Poison, Text Editing, Local.

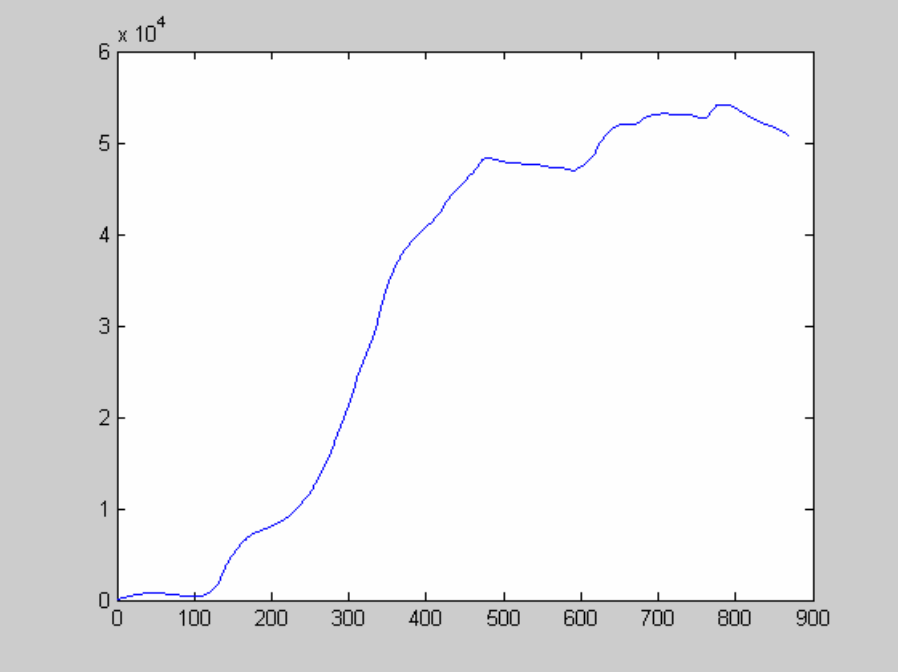


Figure 34. Process(_total)IO other operations/sec, ARP Poison, Text Editing, Remote.

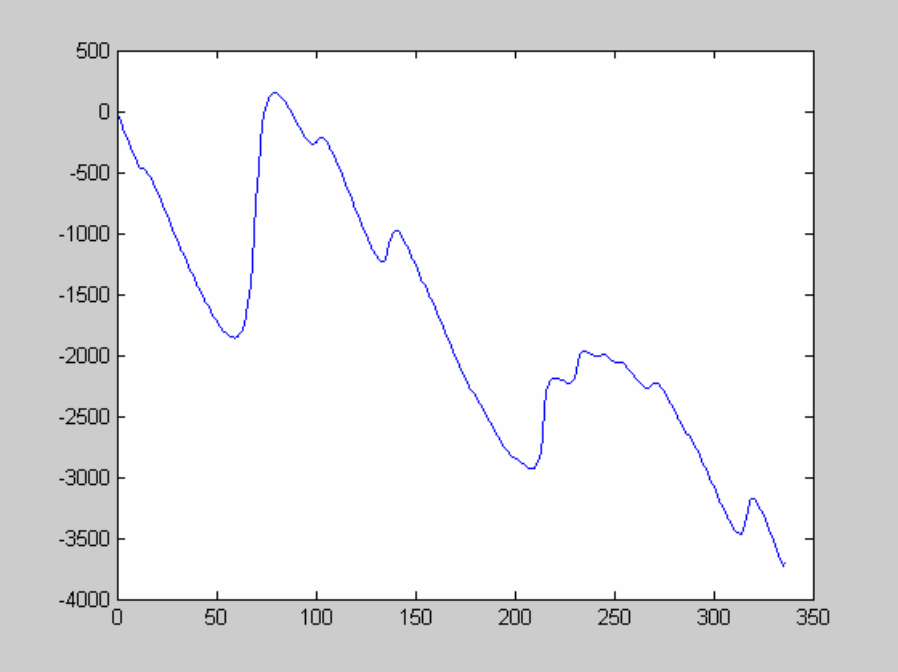


Figure 35. TCP\Segments/sec, EZPublish, Web Browsing, Local.

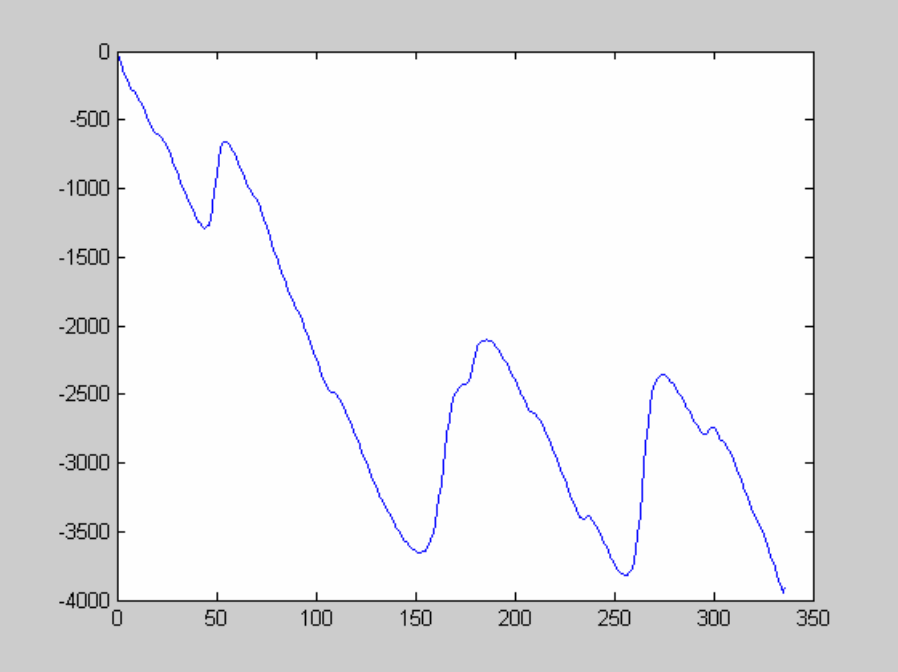


Figure 36. TCP\Segments/sec, EZPublish, Web Browsing, Remote.

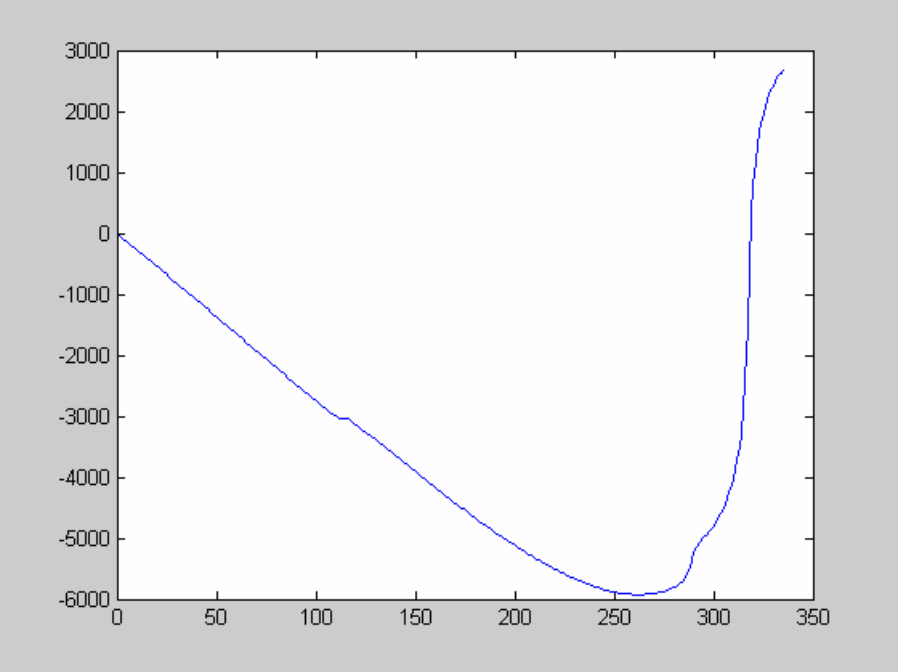


Figure 37. TCP\Segments/sec, EZPublish, Text Editing, Local.

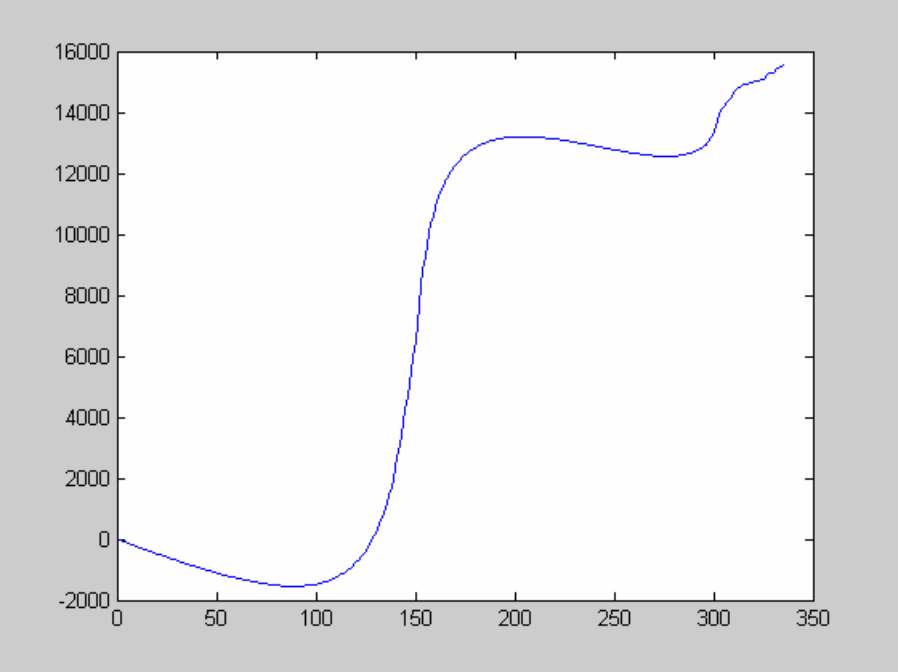


Figure 38. TCP\Segments/sec, EZPublish, Text Editing, Remote.

5.3.1.2 Autocorrelation & Cuscore Statistic

The Cuscore results for this model are shown in Figure 39-Figure 50.

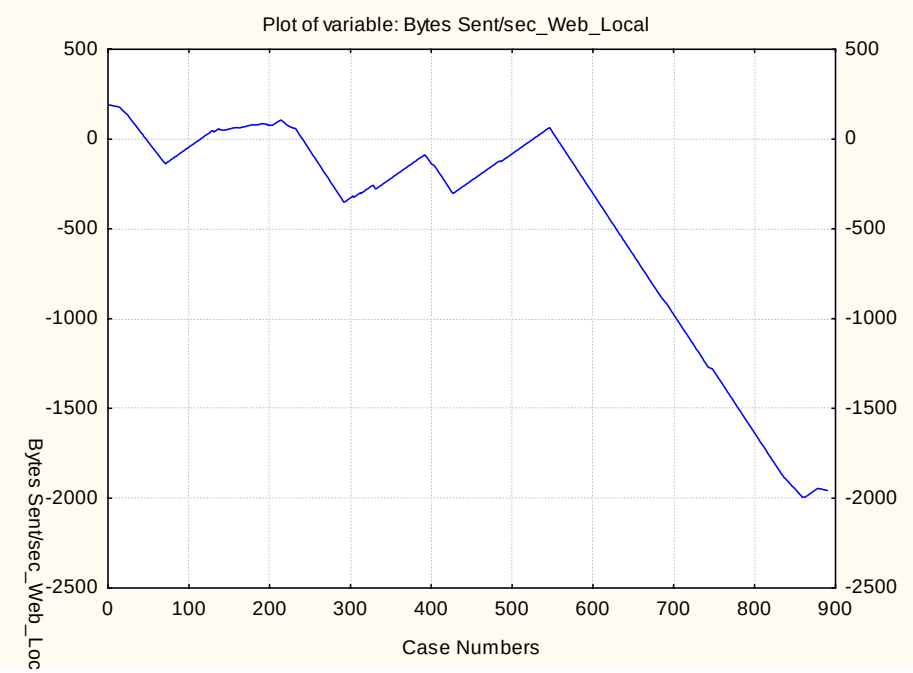


Figure 39. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec, ARP Poison, Web Browsing, Local.

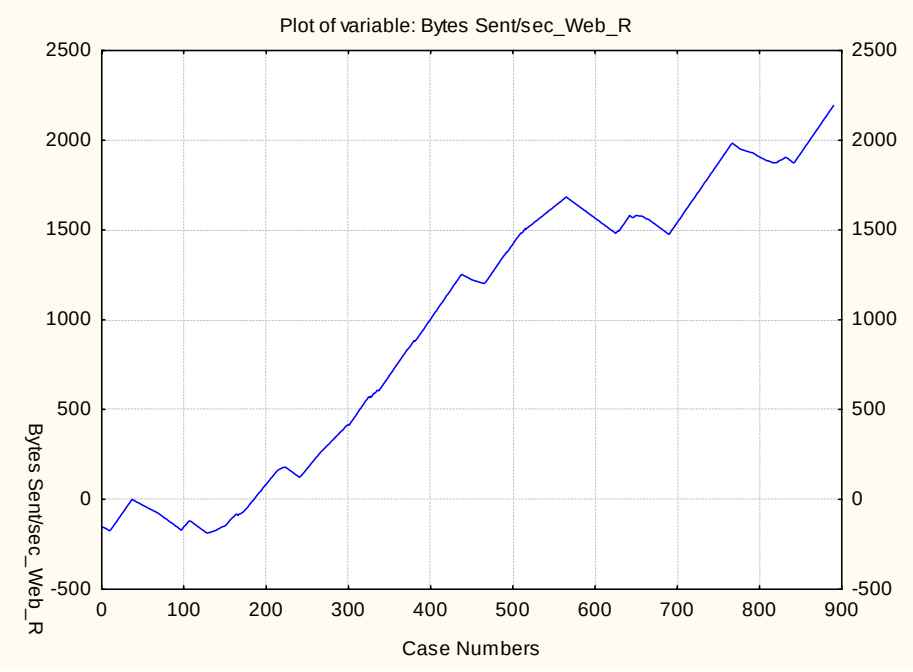


Figure 40. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec, ARP Poison, Web Browsing, Remote.

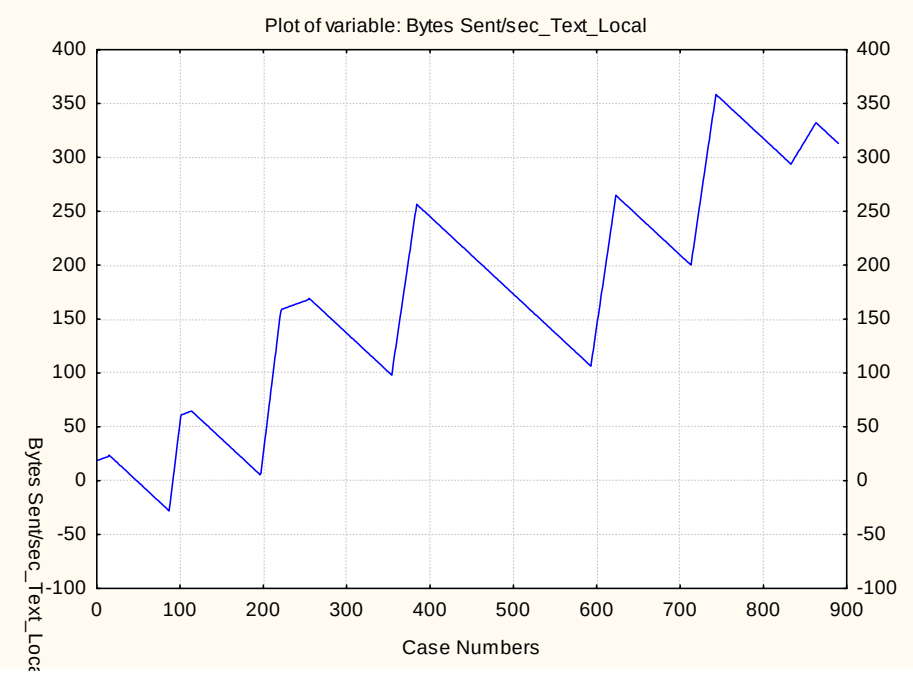


Figure 41. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec, ARP Poison, Text Editing, Local.

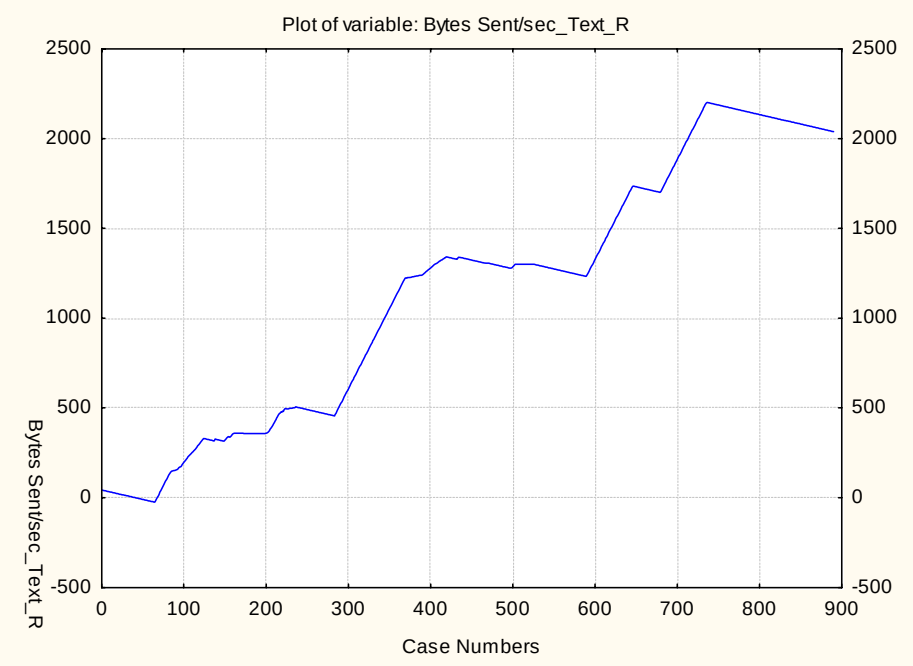


Figure 42. Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec, ARP Poison, Text Editing, Remote

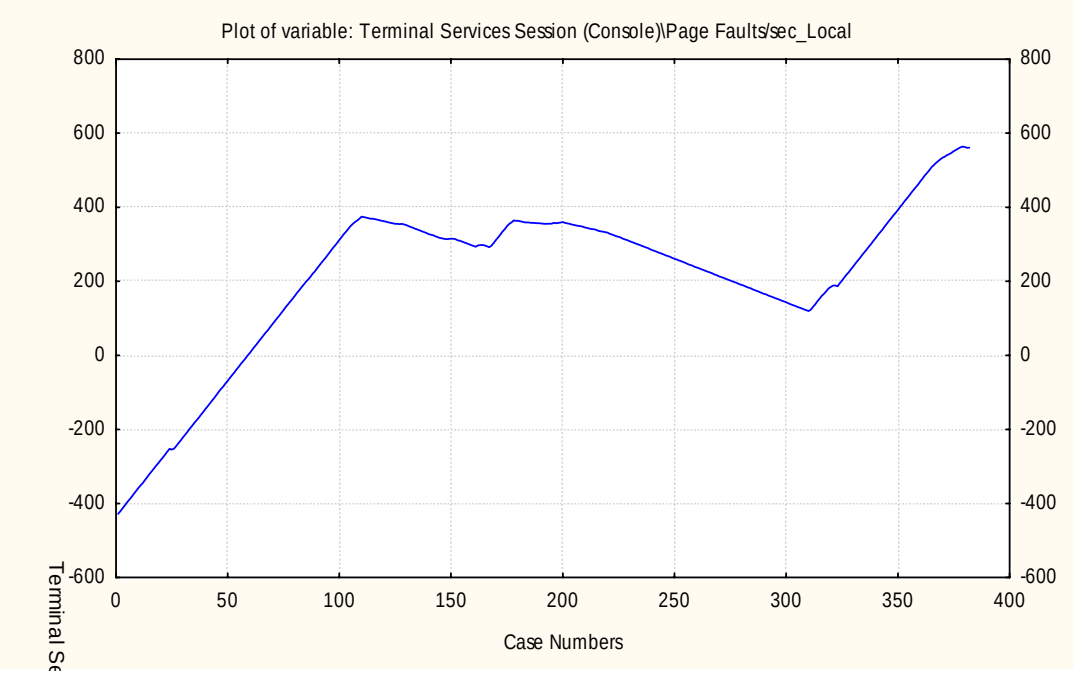


Figure 43. Terminal Services Session(Console)\Page Faults/sec, EZPublish, Web Browsing, Local.

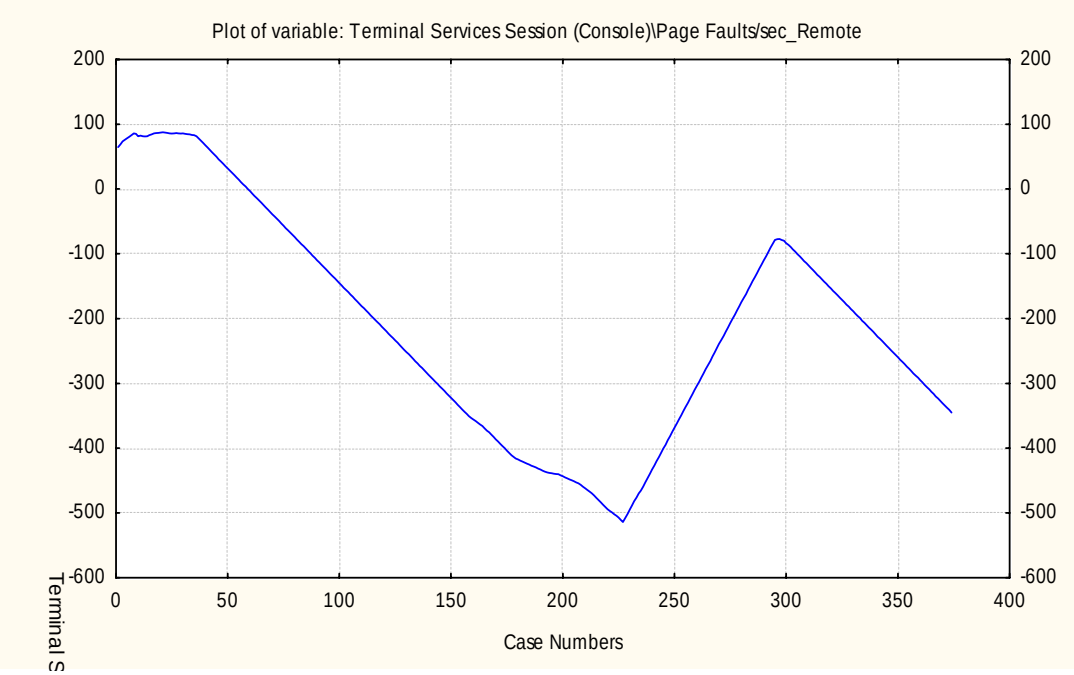


Figure 44. Terminal Services Session(Console)\Page Faults/sec, EZPublish, Web Browsing, Remote.

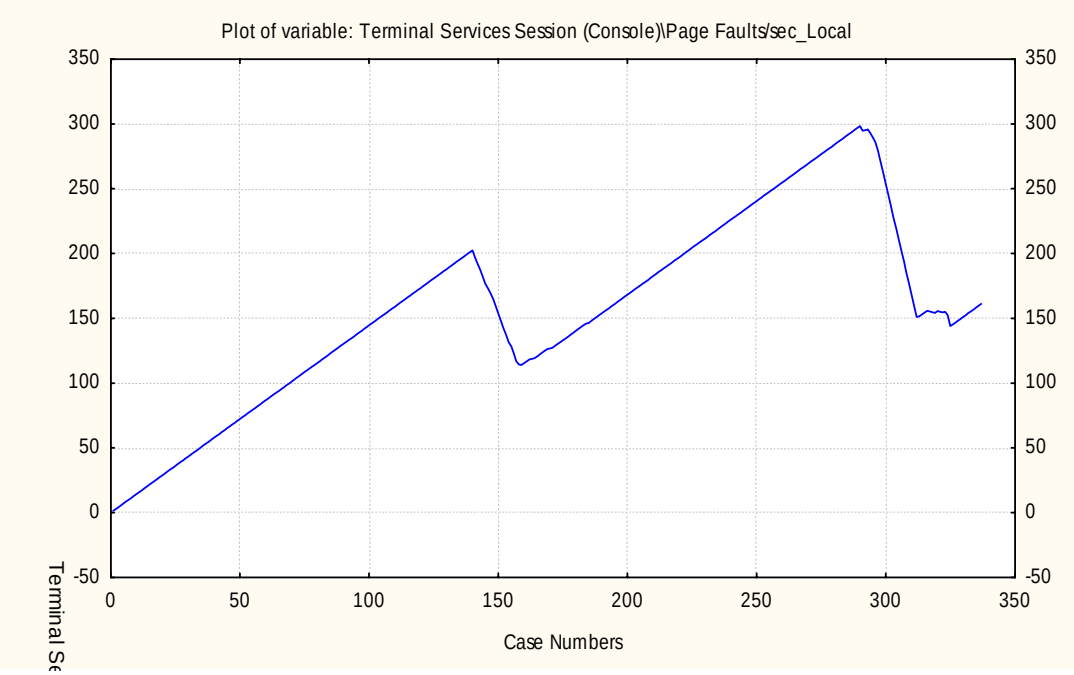


Figure 45. Terminal Services Session (Console)\Page Faults/sec, EZPublish, Text Editing, Local.

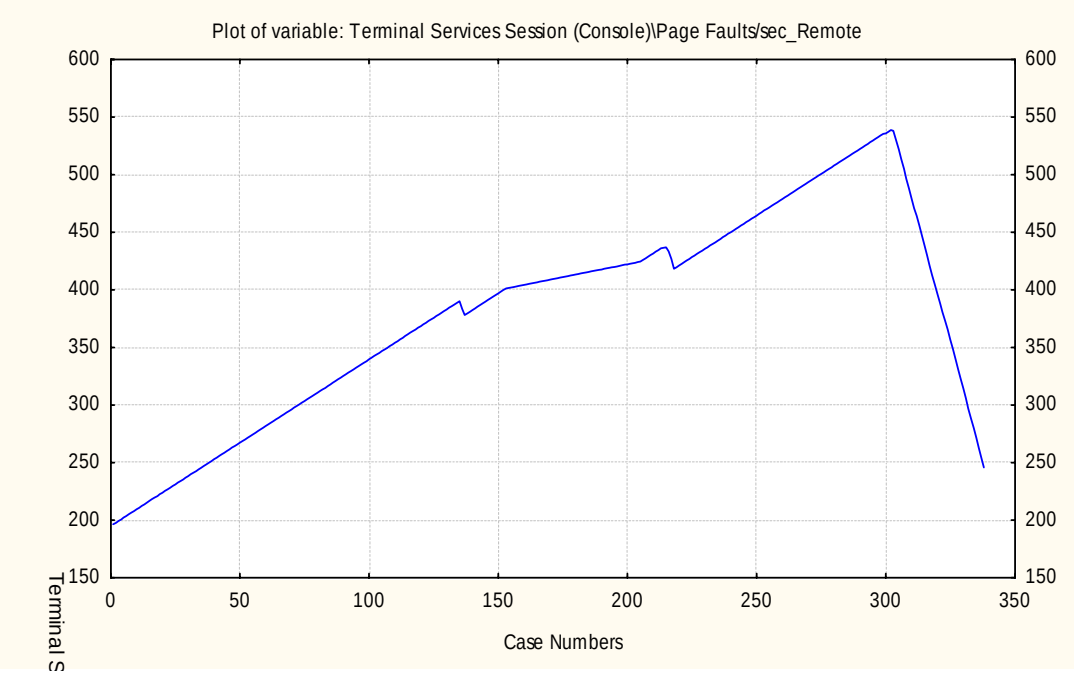


Figure 46. Terminal Services Session (Console)\Page Faults/sec, EZPublish, Text Editing, Remote.

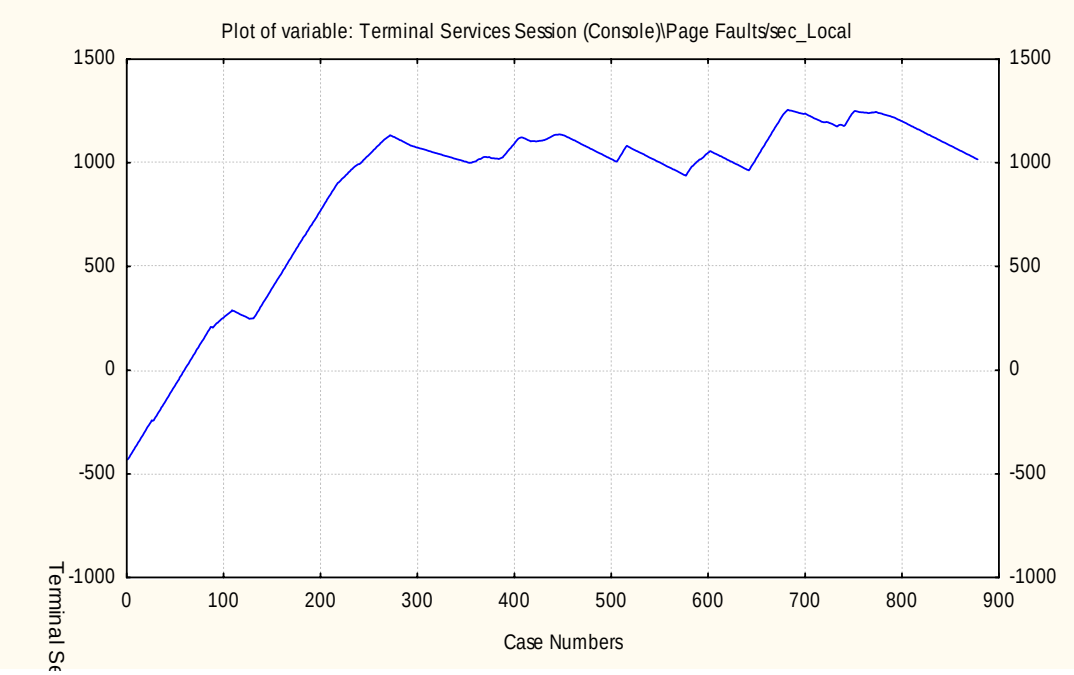


Figure 47. Terminal Services Session (Console)\Page Faults/sec, NMAP, Web Browsing, Local.

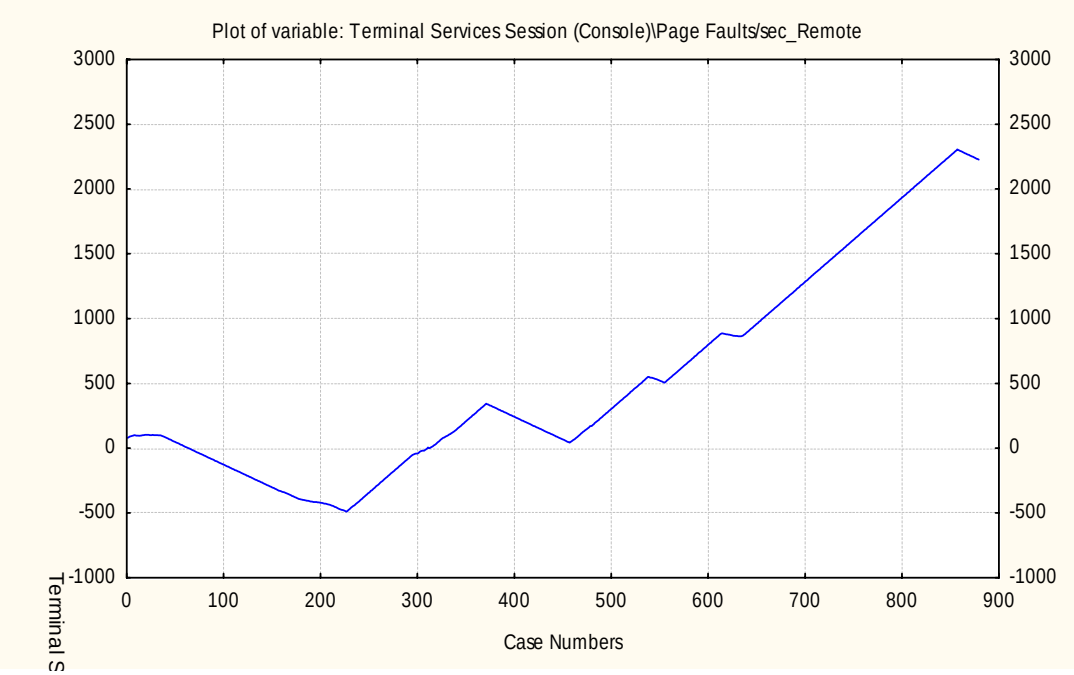


Figure 48. Terminal Services Session (Console)\Page Faults/sec, NMAP, Web Browsing, Remote.

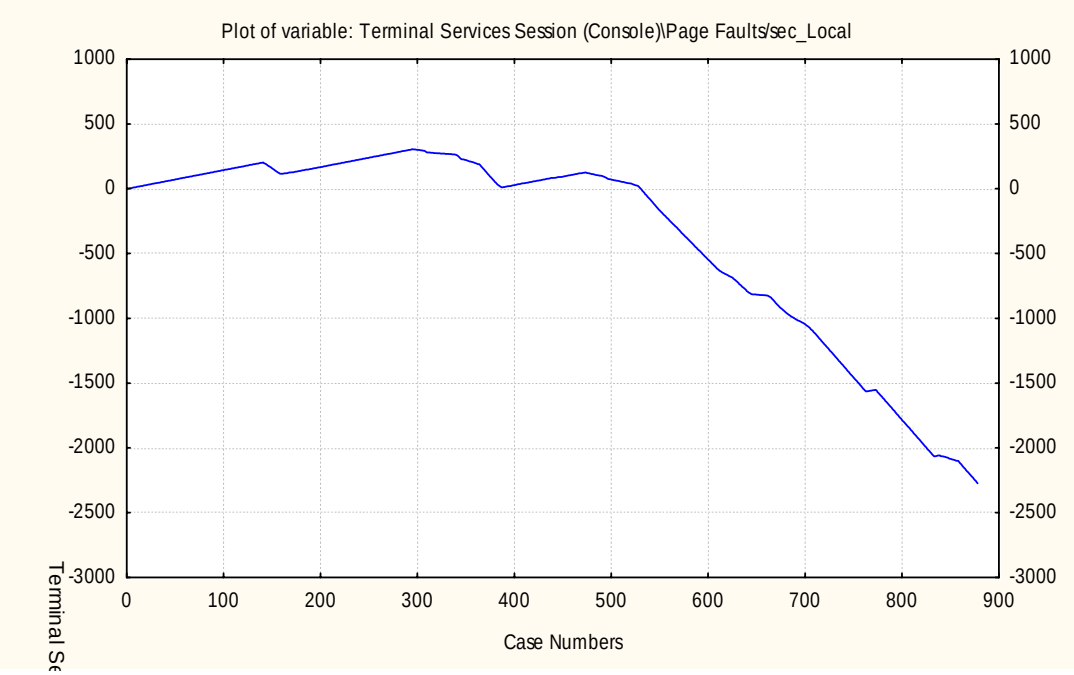


Figure 49. Terminal Services Session (Console)\Page Faults/sec, NMAP, Text Editing, Local.

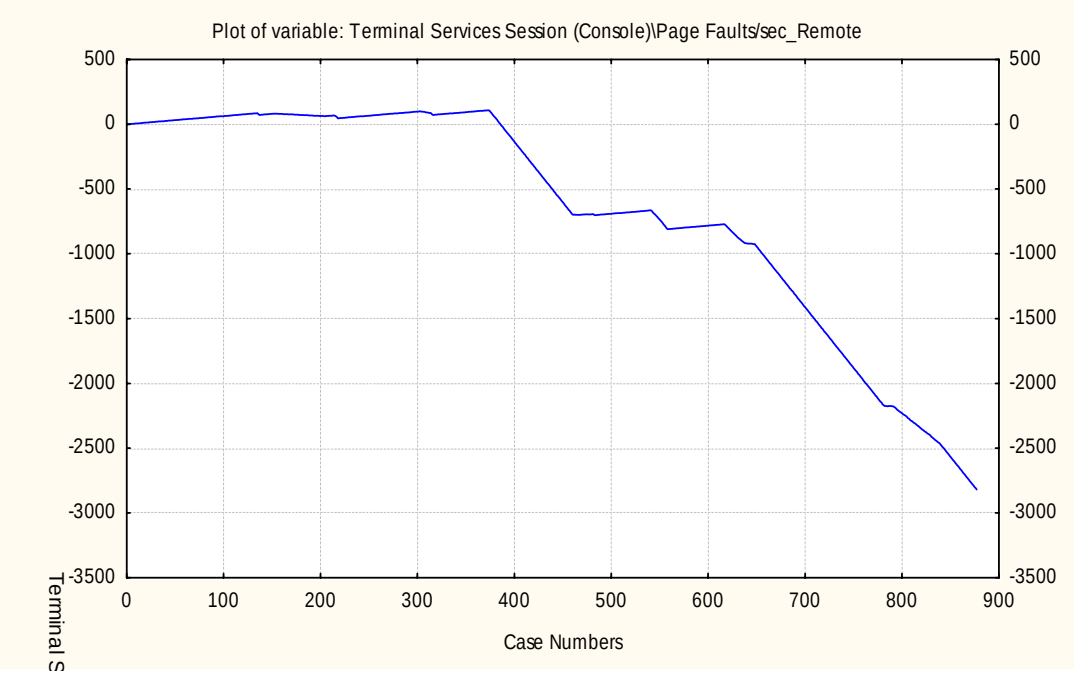


Figure 50. Terminal Services Session (Console)\Page Faults/sec, NMAP, Text Editing, Remote.

5.4 Sensor Fusion

In this section we include the performance summaries of our sensor models in Table 65 and Table 66. From these tables we can make observations for sensor fusion.

Table 65. Paul wavelet and Cuscore statistic

Data Collection →	**Local**		**Remote**	
Attack →	**ARP**	**EZPublish**	**ARP**	**EZPublish**
Variable →	**Process**	**TCP**	**Process**	**TCP**
Web browsing				
False alarm indications	0	5	2	6
Signal indications	1	2	4	1
First observation of signal	288	312	300	292
Text editing				
False alarm indications	0	0	2	2
Signal indications	1	1	5	2
First observation of signal	288	288	288	288

Table 66. Autocorrelation and Cuscore Statistic

Data Collection →	**Local**			**Remote**		
Attack →	**ARP**	**EZPublish**	**NMAP**	**ARP**	**EZPublish**	**NMAP**
Variable →	**Network**	**Terminal**	**Terminal**	**Network**	**Terminal**	**Terminal**
Web browsing						
False alarm indications	2	3	1	5	2	2
Signal indications	5	1	9	7	1	5
First observation of signal	297	311	295	466	297	313
Text editing						
False alarm indications	6	2	1	3	4	0
Signal indications	8	2	3	6	1	4
First observation of signal	355	291	301	287	301	385

During our study, we observe that network variables are affected by remote data collection. Furthermore, we observe consistently throughout our analytical research that using the local data (as opposed to remote) gives better attack detection for the variables considered in these sensor models. Thus, from the results presented in Table 65 and Table 66 we make some example observations for sensor fusion based on the local data collection results.

Note that for the ARP attack, the wavelet sensor detects the attack at data point 288, whereas the autocorrelation sensor does not detect it until points 297 during web browsing, and 355 during text editing. Thus, under each of these activities, the fused model of these two sensors

would not flag this attack using these variables until times 297 and 355 respectively. Thus, both sensors must observe the attack signal for detection.

For the EZPublish attack autocorrelation is the early detector during web browsing and wavelet during text editing. Thus, the earliest attack detection time for this attack, based on the last sensor detection, is 312 for web browsing and 291 for text editing.

The NMAP attack has only one sensor in this optimized model set, and thus there is no sensor fusion for this attack alone.

5.5 Conclusion

In this section, we have extended our analytical discovery results to develop sensor models for cyber attack detection. We test these models under 2 user activity and 2 data collection method conditions. The sensor models we design are based on the sensor optimization section. These sensors are the minimum required to detect and distinguish between the 3 attacks presented in this section. The sensor models we provide are merely samples. There are many other possible models we can develop and test based on our analytical discovery. These methods of sensor model development and fusion can be extended to include any attack or user activity for which we have identified DFCs to identify that activity.

6. Optimized Suite of I&W Observables/Cyber Sensors

In a parallel study, we build sensor models for cyber attack detection. To obtain efficiency, we desire to create the minimum number of sensors possible while maintaining effective attack detection. We employ optimization techniques from operations research to determine the sensor set. This section gives an example for sensor optimization using the analytical discovery results from our previous study on cyber attack characteristics.

6.1 Sensor Matrix

To develop an optimized suite of cyber sensors, we define a sensor matrix. An example sensor matrix is shown in Table 67. In this matrix we used data from 7 data sets to include 6 attacks as described in our previous sections, and one data set provided by an external source.

Table 67. Matrix of Cyber Sensors

	NMAP	Meteor FTP	EZ Publish	IRC Chat	ARP Poison	Netbus Trojan	Nong3
Cache\Data Flushes/sec			(P,-,L) (P,-,M) (P,-,H) (D,-,L) (D,-,M) (D,-,H)			Diff(+)	
Cache\Async Copy Reads/sec				LH-	LH-		-L-
IP\Datagrams/sec	Diff(+) (P,-,L) (D,-,L)		(P,-,L) (P,-,M) (P,-,H) (D,-,L) (D,-,M) (D,-,H)				
Memory\Page Reads/sec				LH- (P,-,L) (D,-,L)	(P,+,L) (D,+,L)		-H-
Memory\Page Writes/sec						(P,+,L) (D,+,L)	-HH
Memory\Available Bytes					Diff(-) -UniUni	Diff(-)	
Memory\% Committed Bytes In Use			-Normal- (P,-,L)		-UniUni (P,-,L) (D,+,L)		
Memory\Cache Faults/sec			-Normal- (P,-,L)			Diff(+) LHL	
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec				LHL (P,-,L) (D,-,L) (D,-,M)	LHL (P,+,M) (D,+,H)	(P,-,L) (D,-,L)	Absent

Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Packets Received Unicast/sec	Diff(+) (P,-,L) (D,-,L)	(P,-,L) (D,-,L)			(P,+,M) (D,+,H)		
Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Total/sec		(P,-,L) (P,-,M) (P,-,H) (D,-,L)		(P,-,L) (D,-,L) (D,-,M)			
Objects\Events		Diff(+) HLH					Diff(+)
Process(_Total)\Page File Bytes			-Normal- (P,-,L) (D,-,L)		-UniUni (P,-,L) (D,+,L)		
Process(EXPLORER)\IO Read Bytes/sec				LH-	LH-		
Process(Meteor#1)\IO Other Operations/sec	Diff(+)	LHL					
Process(Meteor#1)\Handle Count	Diff(+)	Diff(+)					
Processor(_Total)\% Processor Time	Diff(+)		Diff(+) (P,-,L)				
System\File Read Operations/sec	LHL					Diff(+)	
System\System Calls/sec		Diff(+) (P,-,L) (P,-,M) (P,-,H) (D,+,M) (D,+,H)	(P,-,L) (P,-,M) (P,-,H) (D,-,L) (D,-,M) (D,-,H)		(P,-,L) (D,+,L)		
TCP\Segments/sec	Diff(+) (D,-,L)			(P,-,L) (D,-,L)		(P,-,L) (D,-,L)	Diff(+)
Terminal Services Session(Console)\Pool Nonpaged Bytes		Diff(+) HLH (P,-,L) (D,-,L)				Diff(+)	

Terminal Services Session(Console)\Input WdFrames				Diff(-) LHH	LHH (P,-,L) (P,-,M) (D,-,L) (D,-,M) (D,-,H)		
Terminal Services Session(Console)\Input Compressed Bytes				Diff(?) LHH (P,-,L) (P,-,M) (D,-,L) (D,-,M)	LHH (P,-,L) (P,-,M) (D,-,L) (D,-,M) (D,-,H)		
Terminal Services Session(Console)\Input Async Frame Error	Diff(-) (P,+,L) (D,-,L)			Diff(+) (P,+,L) (D,+,L)			
Terminal Services Session(Console)\Protocol Bitmap Cache Hits	Diff(+) -Normal-					(P,-,L) (P,-,M) (P,-,H) (D,-,L) (D,-,M) (D,-,H)	
Terminal Services Session(Console)\Page Faults/sec	Diff(-) HLL (P,+,L) (D,-,L)	HL- (P,-,L) (D,-,L)	Diff(-) HLH				
UDP\Datagrams/sec	Diff(+) (D,-,L)	(P,-,L) (P,-,M) (P,-,H) (D,-,L) (D,-,M) (D,-,H)		(P,-,L) (D,-,L)			

To simplify the display of the table, we have shortened the descriptions in the table cells. The characteristic listed in each cell is a characteristic of the variable from column one, and the attack data from row 1. The entries in the matrix shown in Table 67 are defined as follows:

- Entries of the form Diff(+) or Diff(-) represent a positive or negative shift in the difference in means.
- Entries of the form XXX, where X is L, H or "-" represent autocorrelation results where L is low, H is high, and "-" is none.
- Entries of the form XXX, where X is normal, uni or "-" represent distribution results where the distribution is with normal, uniform (uni) or unknown (-).
- Entries of the form (X,+,Y) or (X,-,Y) represent wavelet results where the first element indicates the wavelet (P for Paul, D for DOG), the second element indicates

increase (+) or decrease (-) from pre-attack to attack, and the third element indicates the frequency band of low (L), medium (M) or high (H).

Cells with multiple entries indicate different sensor models that can be used to detect a characteristic for that variable and that attack. When we optimize the table, if one of these sensor "sets" is chosen, we can select which of the sensors to use. These sensors and variables are in no particular order.

6.2 Sensor Optimization Solution

We find the smallest subset of sensors that can uniquely identify the given seven attacks. This is our optimal collection of sensors. In this section we first present the resulting optimized sensor matrix, and then provide a proof that this table is indeed optimal. This subset was discovered using a manual heuristic based on finding a feasible solution and proving its optimality. The method is revealed in the optimality proof at the end of this section. The optimal solution is shown in Table 68. Note that the original input matrix from Table 67 held 28 sensors, while the optimized solution leads to a smallest subset of only 4 sensors.

Table 68. Optimal Solution

Sensor #	Data Variable	NMAP	Meteor FTP	EZ Publish	IRC Chat	ARP Poison	Netbus Trojan	Nong3
1	Process(_total)IO other operations/sec		(P,-,L)			(P,-,L)		(P,-,L)
2	TCP\Segments/sec			(P,-,L)	(P,-,L)		(P,-,L)	
3	Network Interface(Intel[R] PRO_100 VE Network Connection - Packet Scheduler Miniport)\Bytes Sent/sec				LH	LH		
4	Terminal Services Session(Console)\Page Faults/sec	HL	HL	HL				

Proposition 1. The solution in Table 68 is optimal.

Proof: The structure of the given problem is the same as that of the binary identification problem. In the binary identification problem, the minimum number of sensors that could uniquely identify 7 different attacks is 3. ($2^3 - 1 = 7$, since (000) doesn't count). The optimal solution with 3 binary sensors is shown in Table 69.

Table 69. Optimal solution with 3 binary sensors

Sensor #	NMAP	Meteor FTP	EZ Publish	IRC Chat	ARP Poison	Netbus Trojan	Nong3
1	0	0	1	1	1	0	1
2	0	1	0	1	0	1	1
3	1	0	0	0	1	1	1

Notice that in the 3-binary-sensor case each sensor has to identify at least 4 different attacks. However, the maximum number of attacks one type of sensor could possibly identify is 3 in the given matrix. Therefore, the solution with 4 sensors in Table 68 is optimal.

6.3 Conclusion

In this section, we have shown how to define an optimization table to minimize the number of sensors required to detect a subset of attacks. For this small input table, it is trivial to find the optimal set manually. For larger input tables, any number of optimization tools can be employed for this task. Thus, optimization can be applied to any input matrix of sensors, and thus is extensible to any number of attack and activity data characteristics provided to the optimization problem.

7. Symantec Final Report

This final report documents the activities of Symantec Research Labs (Symantec) on the research project of Arizona State University (ASU) for ARDA's Cyber Indications and Warnings program. The purpose of this project is to study techniques and develop technologies that might provide improved indications and warnings (I&W) for cyber attacks.

After this introductory section, this final report is divided into seven sections according to the statement of work between Symantec and ASU, describing the activities of Symantec with respect to our contractual tasks:

- Task 1 – Collect and examine known cyber attack cases and scenarios to develop threat and attack profiles.
- Task 2 – Discover characteristics of cyber signal and noise (attack data and normal data) at each observable point.
- Task 3 – Investigate, develop and test sensor models of signal detection, and a sensor fusion model for each observable point.
- Task 4 – Formulate and solve an optimization problem to select an optimized suite of I&W observables / cyber sensors.
- Task 5 – Test and verify research outcomes using real information infrastructure data that is available at Symantec.
- Task 6 – Provide documents that reflect monthly status and final technical report.
- Task 7 – Participate in project meetings as necessary.

The final section of this report presents a summary and conclusions. The five technical tasks (tasks 1-5) were conducted on two separate investigations, each addressing a different kind of attack in a different target environment from ASU's investigation. ASU developed the cyber signal detection approach to predictive analysis and investigated that approach. Symantec's first investigation employed ASU's cyber signal detection concepts to explore insider attacks against databases. Symantec's second investigation employed ASU's cyber signal detection approach to explore virus/worm attacks in a local area network. The database investigation was focused on insider attacks at the application layer.

Symantec sought to utilize as much as possible the procedures and research outcomes provided by ASU, evaluating those against malicious code at our disposal at Symantec. ASU provided Symantec with their statistical analyses to run against data collected during attack simulation. Thus, results in this section follow the same approach as described in previous sections in this report. The relevant sections can be reviewed for more detail wherever necessary.

7.1 Task 1 – Collect and examine known cyber attack cases and scenarios to develop threat and attack profiles.

Symantec collected and examined known cyber attack cases and scenarios to develop threat and attack profiles in both the database domain and the virus/worm domain. In the database domain, Symantec collected and examined attack scenarios and developed attack profiles for insider attacks reading confidential data. In the virus/worm domain, Symantec collected and examined attack scenarios and developed attack profiles for the Sobig e-mail virus. The remainder of this section provides details regarding those attack scenarios and attack profiles.

7.1.1 Database Attacks

The database investigation began with a broad examination of known cyber attacks in the database domain. Figure 51 shows a collection of attacks, classified according to a taxonomy developed by ASU. The attacks are given mnemonic names and are organized by "vulnerability".

Attack	Action	Target	State Effects	Performance Effects
Configuration				
Extract-admin-data	Read|Steal	Data	Confidentiality	None
Alter-admin-tables	Modify|Delete|Add	Data	All	All
Monitor-other-DB-activities	Eavesdrop	Data|User	Confidentiality|Availability	Timeliness
Privilege-escalation-or-bypass	Bypass	System	All	All
Administrative-DOS	Termination|Execute	System	Availability	Timeliness
Corrupt-stored-procedures	Modify|Delete	System	All	All
Corrupt-native-libraries	Modify|Delete	System	All	All
Attack-os-using-DB-process-privs	Bypass	System	All	All
Install-unauthorized-components	Modify	System	All	All
Alter-admin-activities	Modify|Delete|Add	System	All	All
Corrupt-future-installations	Modify|Delete|Add	System	All	All
Specification				
Probe-read-confidential-data	Probe	System	Availability	Timeliness
Probe-bypass-privilege-system	Probe	System	Availability	Timeliness
Query-flood	Flood	System	Availability	Timeliness
User Trust				
Read-confidential-data	Read	Data	Confidentiality|Availability	Timeliness
Corrupt-data	Delete|Modify|Add	Data	Integrity|Availability	All

All State Effects = 'Confidentiality' | 'Availability' | 'Integrity'
All Performance Effects = 'Timeliness' | 'Precision' | 'Accuracy'
Source of threat = 'Any'
Agency = Most may be either 'Human' or 'Autonomous'
Attack Origin = 'Local' | 'Remote (single source)' | 'Remote (multiple source)'

Figure 51. Collection of attacks for this study

Because of the large number and heterogeneity of database attacks it was necessary to focus the investigation. The "read-confidential-data" attack was selected as a primary area of focus. This attack simply involves a user reading some confidential data for nefarious purposes.

The "read-confidential-data" attack has an infinite number of variations. Following the pattern being used at ASU, the generic "read-confidential-data" attack was explored by creating an attack profile. The purpose of the attack profile is to identify relevant observables and likely features of attacks. Figure 52 shows the cause-effect chain – a graphical representation of activity-state-performance interactions within an attack – of the attack profile for the generic "read-confidential-data" database insider attack.

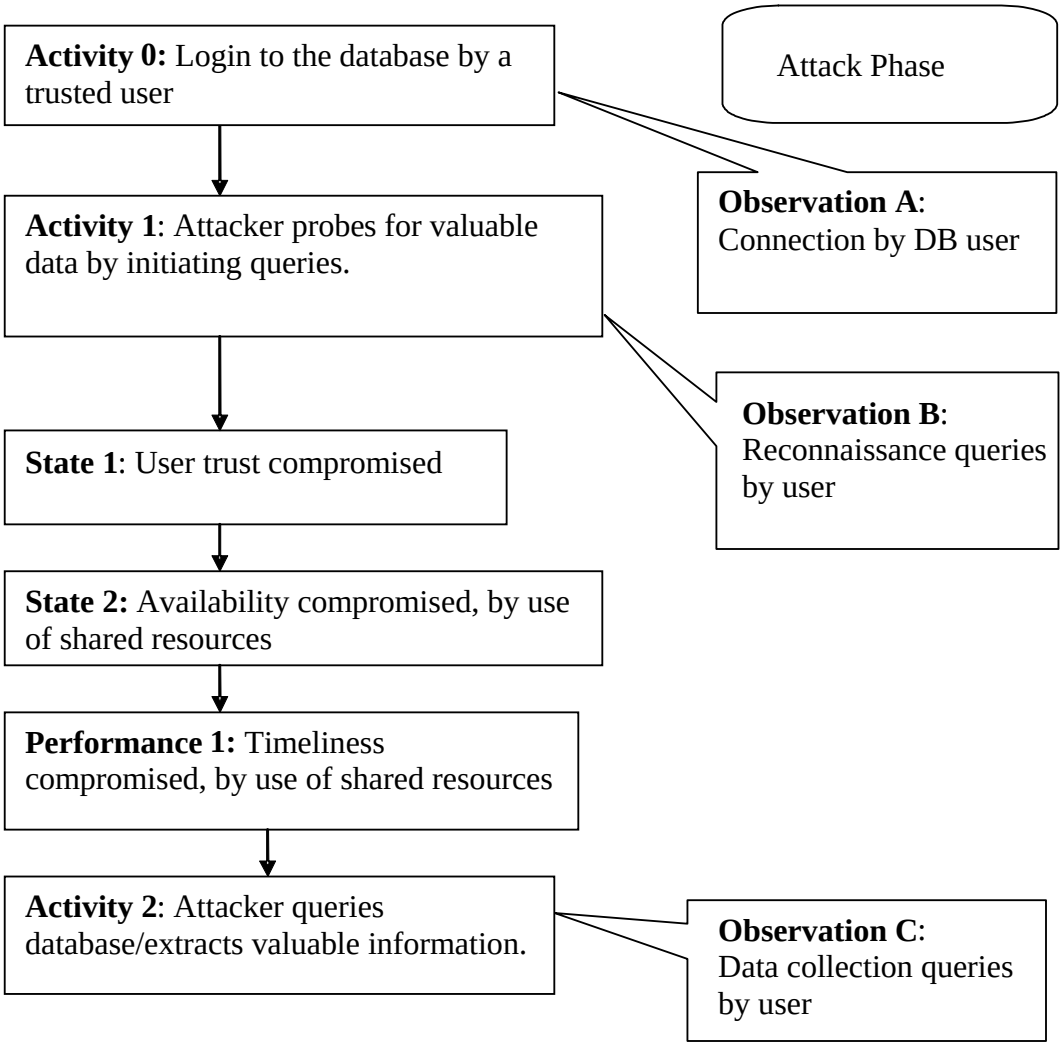


Figure 52. Attack profile for database insider attack

The observations for the generic "read-confidential data" database insider attack are shown in Table 70.

Table 70. Observations for database insider attack

A	Connection by database user
B	Reconnaissance queries by user
C	Data collection queries by user

The attack profile – including DFCs for each observable point – is provided in Table 71.

Table 71. DFC's for database confidentiality attack

OBS	Indicator	Data	Feature	Characteristic
A	1	Time of connection and *database user ID*	Individual observation	Outside historical range
	2	EWMA of duration of connection/session by user	Chi-squared distance	Step change
B	3	Number of tables outside historic range accessed by user within time interval t	Individual observation	Greater than <threshold>
	4	EWMA of InterArrival Time of queries from the same user	Chi-squared distance	Step change
	5	EWMA of queries selecting no data within time interval t (i.e., requested data not found)	Chi-squared distance	Step change
	6	EWMA of size of query text	Chi-squared distance	Step change
	7	EWMA of ratio of SELECT queries to other queries within time interval t	Chi-squared distance	Step change
	8	EWMA of pairwise semantic distances between relations and attributes of successive queries	Chi-squared distance	Step change
	9	EWMA of Number of queries using stored procedure in time interval t	Chi-squared distance	Step change
	10	EWMA of Number of SQL constructs used in queries in time interval t	Chi-squared distance	Step change
C	11	Number of tables outside historic range accessed by user within time interval t	Individual observation	Below <threshold>
	12	EWMA of Interarrival times of queries to same table	Chi-squared distance	Step change
	13	EWMA of Number of queries using stored procedure in time interval t	Chi-squared distance	Step change
	14	Ratio of rows retrieved during session to total rows in table	Mean	Increase
	15	Ratio of columns retrieved during session to total columns in table	Mean	Increase
	16	EWMA of Interarrival times of queries	Chi-squared distance	Step change
	17	EWMA of pairwise semantic distances between relations and attributes of successive queries	Chi-squared distance	Step change
	18	EWMA of number of SQL constructs	Chi-squared distance	Step change

There is no data dependent relationship amongst the observations. The attack formula for the generic "read-confidential-data" database insider attack is the following:

$A(ti\ldots j, l1) \rightarrow B(tk\ldots m, l1) \rightarrow C(tn...p, l1)$, where $i<j<k<m<n<p$ and $l1 = host$

7.1.2 Virus/Worm Attacks

There are many known cyber attack cases in the virus/worm domain. Symantec as a corporation has studied a large percentage of known examples of malicious code and produces descriptions for many of them, available at www.symantec.com. Each malicious code attack has specific characteristics, observables, and effects worthy of study, but clearly our investigation could not address all such attacks. Symantec chose a representative instance of one large class of malicious code attacks, e-mail viruses, for our virus/worm investigation.

Sobig is an e-mail virus from 2003. There were multiple variants of Sobig over the course of that year; the particular variant that Symantec studied was Sobig.f. Like many e-mail viruses, Sobig arrived in a user mailbox as an executable attachment within an infected e-mail. When a user ran the executable attachment, the virus would search the hard drive for e-mail addresses and e-mail a copy of itself to those new addresses.

The observations for the Sobig e-mail virus are shown in Table 72.

Table 72. Observations for the Sobig e-mail virus

A	E-mail infected with Sobig worm received in user's mailbox
B	New process, the Sobig worm, started by user
C	New files created by worm
D	New values added to the startup keys of the registry
E	New worm process starts and original worm process is terminated
F	New event created
G	New threads created
H	Higher CPU utilization as threads search for e-mail addresses
I	Higher file system activity as threads search for e-mail addresses
J	Increased network activity as threads send out infected e-mails
K	Mail activity at mail server as infected e-mails arrives
L	UDP packets initiating download sent to update servers

The Table 73 – including DFCs for each observable point – is provided below.

Table 73. DFC's for Sobig

Obs	Location	Data	Feature	Characteristic
A	L1	New email in the user's inbox (not monitored currently)	Individual observation	9 possible subject lines, 9 possible attachment file names, and 2 possible body lines
	L4	New email, with destination email address of victim	Individual observation	9 possible subject lines, 9 possible attachment file names, and 2 possible body lines
B	L1	Name of new process created	Individual observation	Has 9 possible names
C	L1	Filename of newly created file	Individual observation	"%Windir%\ winppr32.exe"
D	L1	Value of data added to registry key HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion\ Run	Individual observation	"TrayX"="%Windir%\ winppr32.exe /sinc"
E	L1	Name of old process terminated	Individual observation	9 possible names
	L1	Name of new process running	Individual observation	"winppr32.exe"
F	L1	Value of event created	Individual observation	"TrayX"
G	L1	Process object -> thread count for winppr.exe process	EWMA	Step increase
H	L1	Process object -> % processor time for winppr.exe process	EWMA	Step increase
	L1	Filenames of files accessed on disk	Type of file accessed	One of ".dbx, .eml, .hlp, .htm, .html, .mht, .wab, .txt"
I	L1	Process object -> IO data bytes/sec for process winppr.exe	EWMA	Increase
J	L1	IP packets sent/sec from performance log	EWMA	Step increase
	L2	IP packets received/sec and sent/sec from the router's log	EWMA	Step increase
	L4	IP packets received/sec from performance log	EWMA	Step increase
K	L4	E-mails received/sec	EWMA	Step increase
L	L1	DEST IP field of UDP packets, with SRC_port = 8998	Count of unique values	Equals 20

The attack formula for the generic "read-confidential-data" database insider attack is the following:

A (ti, l1) || A (ti, l4) -> B (tj, l1) $\rightarrow$ C (tj..k, l1), D(tj..k, l1), E(tm, l1)
$\rightarrow$ F (tm..n, l1), G (tm..n, l1), H (tn..p, l1), I (tn..p, l1), L (tq..r, l1)

$\rightarrow$ J (tq..r, l1) || J (tq..r, l2) || J (tq..r, l4) $\rightarrow$ K (tq..r, l4)
where l1 = host, l2 = router of victim network, l4 = mail server and i<j<k<m<n<p<q<r

7.2 Task 2 – Discover characteristics of cyber signal and noise (attack data and normal data) at each observable point.

Symantec discovered characteristics of cyber signal and noise in the virus/worm domain for the Sobig e-mail virus at the following observable points:

- the attacker machine's performance variable monitor
- the victim machine's performance variable monitor
- the bystander machine's performance variable monitor
- the mailserver machine's performance variable monitor

The characteristics of cyber signal and noise that we discovered for the Sobig e-mail virus include the following:

There is a baseline of pairwise Pearson correlation between variables, even when there is no user activity, as demonstrated by the pre-attack phase on each machine during every experiment. The non-zero, non-invaried performance variables are likely caused by operating system and services' process activity. The pairwise correlation amongst those performance variables is likely because some or all the variables for each performance object are pairwise correlated independent of the activity on the machine. This baseline activity and correlation is a form of cyber noise with respect to the performance variable observable point.

In the experiment with only attack activity, on all four machines the percentage of pairwise correlated variables decreases from the pre-attack to the attack phase as attack-related activity occurs. This is most surprising on the bystander machine, which should be relatively unaffected by the attack activity. Nevertheless, this indicates that cyber signal creates a drop in pairwise correlation among performance variables for the Sobig e-mail virus experiment.

Going from the attack to the post-attack phase in the experiment with only attack activity, on the attacker and mail server machines there is a slight increase in pairwise correlation percentage, and on the victim machines there is a slight decrease. The attacker and mail server machines will have more attack activity than the bystander and victim machines again indicating that stronger cyber signal has an effect on the percentage of pairwise correlation among performance variables in the case of the Sobig e-mail virus.

In the experiments with attack and normal data (FTP and text editing user activity), the percentage of pairwise correlated variables on the victim machine is lower during the attack phase (with user activity continuing concurrently) than in the pre-attack with user activity phase. These results are similar to the results on the victim machine in the experiment with only attack data. The characteristic of cyber signal that is confirmed is that attack activity lowers the percentage of correlated performance variables, for variables that are non-zero and non-invaried within any given phase, in the case of the Sobig e-mail virus. (The results when using the same set of non-zero, non-invaried variables across all phases are different in the text editing experiment.)

In the text editing experiment, the post-attack phase has the smallest correlation percentage of any of the five phases on the victim machine. For the FTP experiment, it has the least number of significant correlations. The distinction between "smallest correlation

percentage" and "least number of significant correlations" is that "correlation percentage" is the "number of significant correlations" divided by the total cells (the number of comparisons possible between non-zero, non-invaried variables). The "number of significant correlations" is determined by the Pearson correlation analysis. The "total cells" is determined in a pre-analysis stage when performance variables that are all zeroes or all the same value are thrown out, leaving only the non-zero, non-invaried variables for Pearson correlation pairwise comparisons. Therefore, the "correlation percentage" divides the "number of significant correlations" by the "total cells" to calculate a percentage, versus simply looking at the raw number of significant correlations.

- With the start of user activity in the text editing experiment, there is a decrease in the correlation percentage from the pre-attack stage on the victim machine. In the FTP experiment though, there is a slight increase in correlation percentage when user activity starts. This is likely caused by the difference in activity type and its effect on performance variables – highly network-intensive user activity appears to be more correlated.
- The post-attack with user activity phase on the victim machine in the FTP experiment exhibits a similar correlation percentage to the pre-attack with user activity phase, after dipping during the attack phase. This seems to demonstrate again that the form of cyber noise generated by network-intensive user activity is relatively high when compared to other phases of the experiment, and it is unaffected by the aftereffects of the Sobig e-mail virus.
- On the victim machine, across all three experiments (only attack activity, attack and text editing user activity, attack and FTP user activity), the number of non-zero, non-invaried variables common to both local and remote data collection methods is highest during the attack phase. This also holds true on both the attacker and mail server machines in the experiment with only attack activity. This is an important characteristic of cyber signal for this worm: attack activity consistently increases the number of non-zero, non-invaried performance variables for the Sobig e-mail virus. In other words, attack activity increases the number of relevant performance variables, which are those that are non-zero and non-invaried, both with and without normal activity, and on all machines involved in the attack (i.e., all except the bystander).

As mentioned previously, for the virus/worm attacks we utilize ASU's experimental process and analysis code for discovering characteristics of cyber signal and noise. Symantec applied a subset of the analyses to performance variable data from all four machines in our Sobig experiments. We provide an overview of experimental process and analysis code, and how they map to the Sobig attack and our laboratory environment, for more details please refer to the ASU technical reports.

The next subsection describes the experimental process Symantec utilized for discovering characteristics of cyber signal and noise. Following that, we present the results from experimentation with (only) attack data, and then from experimentation with both attack and normal data.

7.2.1 Experimental Process

For each of the observable points (machines), analysis code from ASU's cyber signal detection approach was run against the data to discover characteristics of cyber signal and noise.

The performance variable monitor on Microsoft Windows machines collects detailed, low-level information regarding the performance of hardware and operating system components (called objects within the performance monitor). The performance variable monitor can be accessed on Windows XP through the Start Menu: Programs → Administrative Tools → Performance. Example performance objects include the processor, memory, hard disk, network interface, process scheduler, and threads, and for each object there are many variables. On a typical machine, there can be between 3,500 and 4,000 variables spread across approximately 30 objects. For a machine configured as a mail server, there are approximately 5,000 performance variables.

One important component of the experimental process is that each experiment was run twice, using a different method of collection for the performance variables. The first method of collecting the performance variables occurred on each experimental machine itself. This method is called local data collection. The second method of collecting the performance variables occurred on a remote machine by transmitting them over the network. This method is called remote data collection. Using these two methods enabled us to factor out some of the effects of the data collection process on the experiment, as the sheer volume of data collection undoubtedly has an impact on the observed machine. We present the results of both the local and remote data collection experiments in each subsection below.

The first set of analyses screens for variables with all zeroes and variables with all of the same values. Screening those variables out leaves remaining only the non-zero, non-invaried variables – those variables that could be statistically significant for the experiment. After the non-zero, non-invaried variables are determined for each machine during each phase of the experiment, we calculate the common significant variables between the local and remote phases. This helps to remove some of the effect of the data collection method: any variable that is statistically significant only in one of the two data collection methods – local or remote – is likely demonstrating some artifact of the data collection. We present the results of this first set of analyses because there are a great number of variables that are screened out in this first stage, and that number varies greatly by observable point (machine) and by phase of the experiment.

The second analysis provided by ASU is Pearson correlation. Pearson correlation is a measure of correlation between two variables. In the tables presenting our experimental results, the total number of pairwise comparisons (called Total Cells) is a function of the number of non-zero, non-invaried variables – each variable is compared against every other one. We utilize only the common non-zero, non-invaried variables between the local and remote data collection methods for each machine during each phase of the experiment when calculating the Pearson correlation. We then present the results for the number and percentage of correlated variables (for both data collection methods).

7.2.2 Experimentation with Attack Data

For experimentation with only attack data, ASU divided each experiment into three phases: pre-attack (10 minutes), attack (approximately 3 minutes for Sobig), and post-attack (10 minutes). The pre-attack phase was a period of inactivity to establish a baseline of performance variable data for the experiment. The attack phase encompassed the amount of time from when the attack was launched to when it was completed and/or terminated. In the case of Sobig, this phase starts when the virus is run on the attacker machine; the infected attacker machine then sends e-mail to the victim machine, where it is received, opened, and executed; the attack phase is terminated by killing the virus process on both the attacker and victim machines. The post-attack phase collects data on any aftereffects of the attack, with no activity taking place.

Table 74 presents the number of non-zero, non-invaried performance log variables on the attacker machine. The measures are derived from performance variables that are installed by default on commonly deployed operating systems (i.e., Windows 2000 and XP), as described previously in this report. The purpose of this investigation was exploration of possible solution strategies.

Table 74. Non-zero, non-invaried performance log variables on the attacker machine

Phase	Local	Remote	Common
Pre-attack	849	693	576
Attack	850	801	683
Post-attack	899	802	660

Table 75 presents the number of non-zero, non-invaried performance log variables on the bystander machine.

Table 75. Non-zero, non-invaried performance log variables on the bystander machine

Phase	Local	Remote	Common
Pre-attack	911	674	617
Attack	599	624	464
Post-attack	733	675	582

Table 76 presents the number of non-zero, non-invaried performance log variables on the mail server machine.

Table 76. Non-zero, non-invaried performance log variables on the mail server machine

Phase	Local	Remote	Common
Pre-attack	1031	963	807
Attack	988	1004	876
Post-attack	916	858	748

Table 77 presents the number of non-zero, non-invaried performance log variables on the victim machine.

Table 77. Non-zero, non-invaried performance log variables on the victim machine

Phase	Local	Remote	Common
Pre-attack	827	554	404
Attack	710	722	574
Post-attack	665	529	407

The non-zero, non-invaried screening results above show a substantial reduction in the number of significant performance variables at all four observable points (machines) during all phases of the experiment. (Recall that 3500 to 5000 performance variables are collected for each experiment.) In general, the number of significant variables is higher during local data collection than remote data collection for each machine during the pre-attack and post-attack phases, while the opposite is true for the attack phase. Furthermore, the number of common significant variables is highest in the attack phase for all but the bystander machine. (It is interesting that the bystander machine – which should be relatively unaffected by the attack activity – shows wide variations between phases: especially the large drop in significant variables during the attack phase in local data collection mode.)

Table 78 presents a summary of the results from the Pearson correlation analyses on the attacker machine.

Table 78. Summary results from the Pearson correlation analyses on the attacker machine

Phase	Total Cells	Significant Correlations	Correlation Percentage
Pre-attack (local)	165025	103433	62.68%
Pre-attack (remote)	165025	105353	63.84%
Attack (local)	232221	118963	51.23%
Attack (remote)	232221	132628	57.11%
Post-attack (local)	216811	115598	53.32%
Post-attack (remote)	216811	131359	60.59%

Table 79 presents a summary of the results from the Pearson correlation analyses on the bystander machine.

Table 79. Summary results from the Pearson correlation analyses on the bystander machine

Phase	Total Cells	Significant Correlations	Correlation Percentage
Pre-attack (local)	189420	122899	64.88%
Pre-attack (remote)	189420	117483	62.02%
Attack (local)	106953	52839	49.40%
Attack (remote)	106953	61919	57.89%
Post-attack (local)	168490	88053	52.26%
Post-attack (remote)	168490	80718	47.91%

Table 80 presents a summary of the results from the Pearson correlation analyses on the mail server machine.

Table 80. Summary results from the Pearson correlation analyses on the mail server machine

Phase	Total Cells	Significant Correlations	Correlation Percentage
Pre-attack (local)	324415	202256	62.34%
Pre-attack (remote)	324415	178985	55.17%
Attack (local)	382375	175916	46.01%
Attack (remote)	382375	186074	48.66%
Post-attack (local)	278631	131642	47.25%
Post-attack (remote)	278631	144224	51.76%

Table 81 presents a summary of the results from the Pearson correlation analyses on the victim machine.

Table 81. Summary results from the Pearson correlation analyses on the victim machine

Phase	Total Cells	Significant Correlations	Correlation Percentage
Pre-attack (local)	81003	44479	54.91%
Pre-attack (remote)	81003	44850	55.37%
Attack (local)	163878	77639	47.38%
Attack (remote)	163878	80242	48.96%
Post-attack (local)	82215	32640	39.70%
Post-attack (remote)	82215	37949	46.16%

Figure 53 depicts the correlation percentages across all four machines for both local and remote data collection modes.

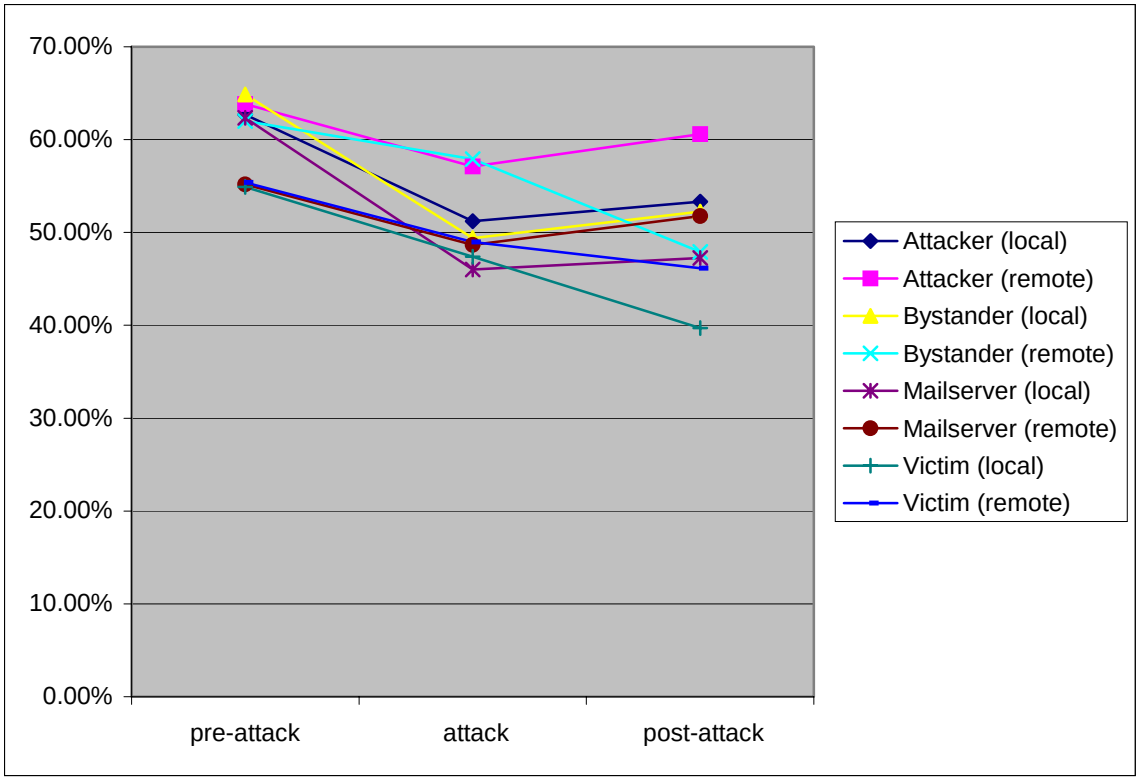


Figure 53. Correlation percentages across four machines

In the Pearson correlation results above, with very few exceptions, the difference in correlation percentage between the local and remote data collection methods is within a few percentage points. This is at most an eight percent differential (bystander machine, attack phase). Given the relative proximity in correlation percentages between the two data collection modes, we will speak generally about each phase of the experiment independent of data collection mode.

The Pearson correlation results above demonstrate the following characteristics of cyber signal:

- On all four machines, the percentage of pairwise correlated variables decreases from the pre-attack to the attack phase as attack-related activity occurs. Again, this is most surprising on the bystander machine, which should be relatively unaffected by the attack activity. Nevertheless, this indicates that cyber signal creates a drop in pairwise correlation among performance variables for the Sobig e-mail virus experiment. The average drop across all four machines is 9.32%. For this reason, given the differences between local and remote data collection mentioned above, the next subsection "Experimentation with Attack and Normal Data" focuses on variables common to both local and remote methods as non-zero and non-invaried.
- Going from the attack to the post-attack phase, on the attacker and mail server machines there is a slight increase (average 2.48% rise) in pairwise correlation percentage. On the victim machine there is a slight decrease (average 5.24% drop).

The attacker and mail server machines will have more attack activity than the bystander and victim machines again indicating that stronger cyber signal has an effect on the percentage of pairwise correlation among performance variables in the case of the Sobig e-mail virus. As mentioned previously, given the differences between local and remote data collection mentioned above, the next subsection "Experimentation with Attack and Normal Data" focuses on variables common to both local and remote methods as non-zero and non-invaried.

The Pearson correlation results demonstrate the following characteristics of cyber noise:

- There is a baseline of pairwise correlation between variables, even when there is no user or attack activity, as demonstrated by the pre-attack phase on each machine. The non-zero, non-invaried performance variables are likely caused by operating system and services' process activity. The pairwise correlation amongst the performance variables is likely because some or all the variables for each performance object are pairwise correlated independent of the activity on the machine. This baseline activity and correlation is a form of cyber noise with respect to the performance variable observable point.

7.2.3 Experimentation with Attack and Normal Data

For experimentation with both attack and normal data, ASU divided each experiment into five phases: pre-attack (10 minutes), pre-attack with user activity (10 minutes), attack (approximately 3 minutes for Sobig), post-attack with user activity (10 minutes), and post-attack (10 minutes). The pre-attack phase and post-attack phases were the same as in the experiments with only attack data: no activities were taking place on the machines at the beginning and end of the experiment, respectively. During the pre-attack with user activity phase, some form of normal data was generated via user activity, prior to the attack phase. The attack phase was performed in the same manner as in the experiments with only attack data, except that the user activity was continued throughout the attack phase in these experiments. Finally, in the post-attack with user activity phase, the attack was terminated but user activities continued.

Symantec completed experiments with two different types of user activity, which correspond to two of the user activity experiments that ASU performed:

- Text editing: a user types text, with figures and tables, into a Word document, saving periodically. The same text that ASU used in their experimentation is used in our experiments. This experiment represents a primarily host-based activity.
- FTP downloading: a user downloads files from an FTP server continuously. ASU utilized e-book files for downloading; the files on our FTP server were a variety of text and binary files of varying sizes. This experiment represents a primarily network-based activity.

Similar to ASU in their experimentation, user activity is only conducted on a single machine – the victim – for our Sobig experiments. For this reason, only data from the victim machine is analyzed for characteristics of cyber signal and noise in the following subsections.

7.2.3.1 User Activity: Text Editing

Table 82 presents the number of non-zero, non-invaried performance log variables on the victim machine.

Table 82. Non-zero, non-invaried performance log variables on the victim machine

Phase	Local	Remote	Common
Pre-attack	715	635	356
Pre-attack w/ user activity	818	696	489
Attack	867	738	529
Post-attack w/ user activity	725	636	416
Post-attack	804	564	409

As in the experiments with only attack data, the number of non-zero, non-invaried variables during each phase of the experiment on the victim machine is higher during local data collection than remote data collection. The number of non-zero, non-invaried variables rises from the pre-attack phase to the pre-attack with user activity and again to the attack phase, then falls in the post-attack with user activity phase and falls again in the post-attack phase. This indicates that as more activity occurs on the victim machine – whether that activity is normal usage or attack activity – the number of significant performance variables increases. This corresponds roughly to the pattern in the experiment with only attack data (with only three phases of the experiment though).

Table 83 presents a summary of the results from the Pearson correlation analyses on the victim machine for the text editing experiment.

Table 83. Pearson correlation analyses on the victim machine for the text editing

Phase	Total Cells	Significant Correlations	Correlation Percentage
Pre-attack (local)	62835	30061	47.84%
Pre-attack (remote)	62835	29469	46.90%
Pre-attack w/ user activity (local)	118828	47720	40.16%
Pre-attack w/ user activity (remote)	118828	50787	42.74%
Attack (local)	139128	59458	42.74%
Attack (remote)	139128	57265	41.16%
Post-attack w/ user activity (local)	85905	29422	34.25%
Post-attack w/ user activity (remote)	85905	38369	44.66%
Post-attack (local)	83028	32936	39.67%
Post-attack (remote)	83028	29533	35.57%

Figure 54 depicts the correlation percentage for both local and remote data collection modes.

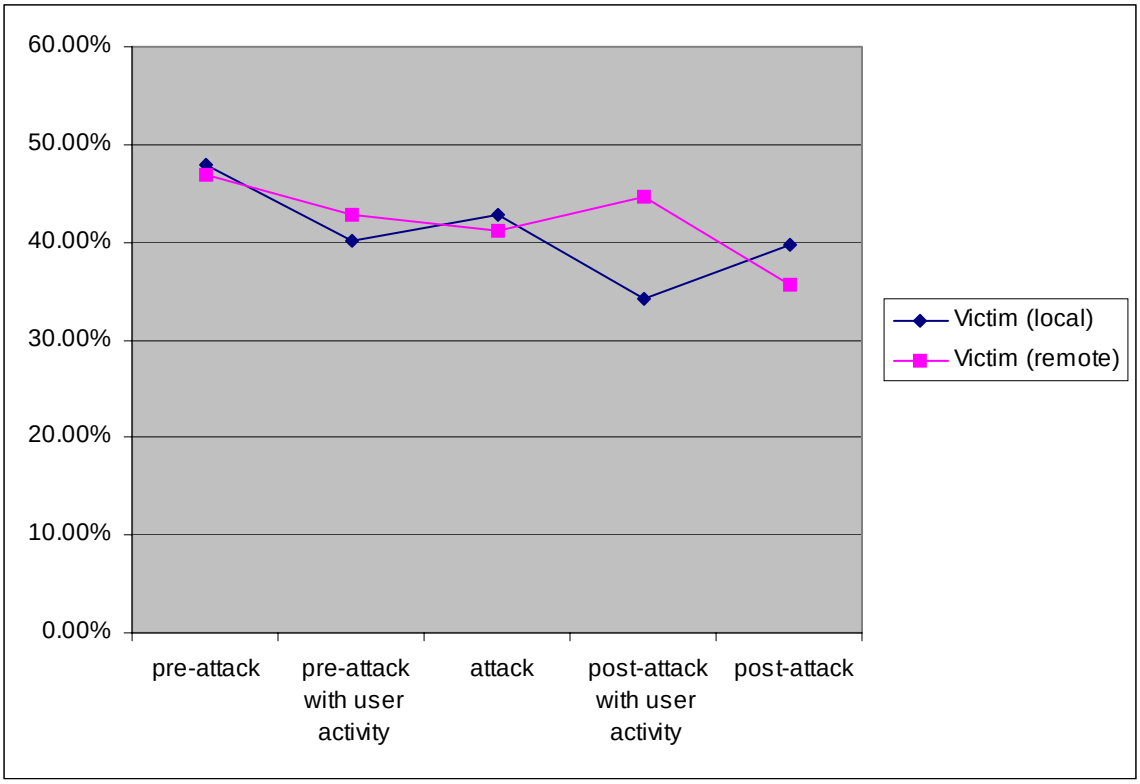


Figure 54. Correlation percentage for both local and remote data collection modes

After screening only the common significant variables and then calculating the Pearson correlation percentages, the difference between local and remote data collection is within a few percentage points (with the exception of the post-attack with user activity phase).

One other interesting and potentially useful view of the data is to study the changes in correlation percentages when the same performance variables are examined across all phases of the experiment. In order to study this scenario, we screened the data from all phases of the experiment in both local and remote data collection modes for only the variables that were always non-zero and non-invaried. For the text editing experiment on the victim machine, this was 252 variables, yielding 31,375 correlation pairs. The Figure 55 below depicts the correlation percentages in both local and remote data collection modes using only the common non-zero, non-invaried variables.

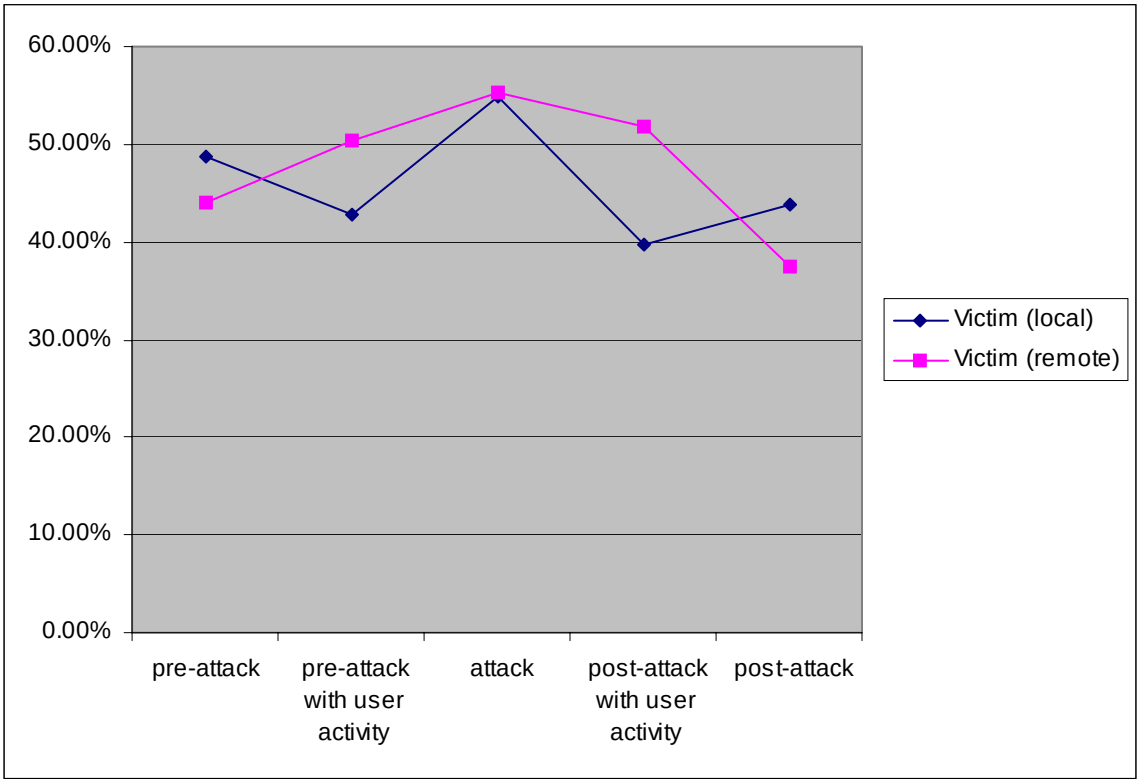


Figure 55. Correlation percentages using only common non-zero, non-invaried variables

Our observations regarding the characteristics of cyber signal and noise will be presented after the next subsection, independent of the differences in local and remote data collection results.

7.2.3.2 User Activity: FTP Downloading

The Table 84 below presents the number of non-zero, non-invaried performance log variables on the victim machine.

Table 84. Non-zero, non-invaried performance log variables on the victim machine

Phase	Local	Remote	Common
Pre-attack	695	808	395
Pre-attack w/ user activity	694	992	507
Attack	708	944	557
Post-attack w/ user activity	706	893	506
Post-attack	434	702	316

In this experiment, the number of non-zero, non-invaried variables during each phase of the experiment on the victim machine is higher during remote data collection than local data collection – this is in contrast to previous experiments. This could be a result of the highly network-intensive user activity. As in the text editing experiment, the number of non-zero, non-invaried variables rises from the pre-attack phase to the pre-attack with user activity and again to the attack phase, then falls in the post-attack with user activity phase and falls again in the post-attack phase. This pattern of common variables across experiment phases is a consistent observation across all experiments on the victim machine.

The Table 85 below presents a summary of the results from the Pearson correlation analyses on the victim machine for the FTP downloading experiment.

Table 85. Pearson correlation analyses on the victim machine for FTP downloading

Phase	Total Cells	Significant Correlations	Correlation Percentage
Pre-attack (local)	77421	32136	41.51%
Pre-attack (remote)	77421	34407	44.44%
Pre-attack w/ user activity (local)	127765	58907	46.11%
Pre-attack w/ user activity (remote)	127765	61481	48.12%
Attack (local)	154290	54743	35.48%
Attack (remote)	154290	59629	38.65%
Post-attack w/ user activity (local)	127260	57904	45.50%
Post-attack w/ user activity (remote)	127260	60767	47.75%
Post-attack (local)	49455	13562	27.42%
Post-attack (remote)	49455	20200	40.85%

Figure 56 depicts the correlation percentage for both local and remote data collection modes.

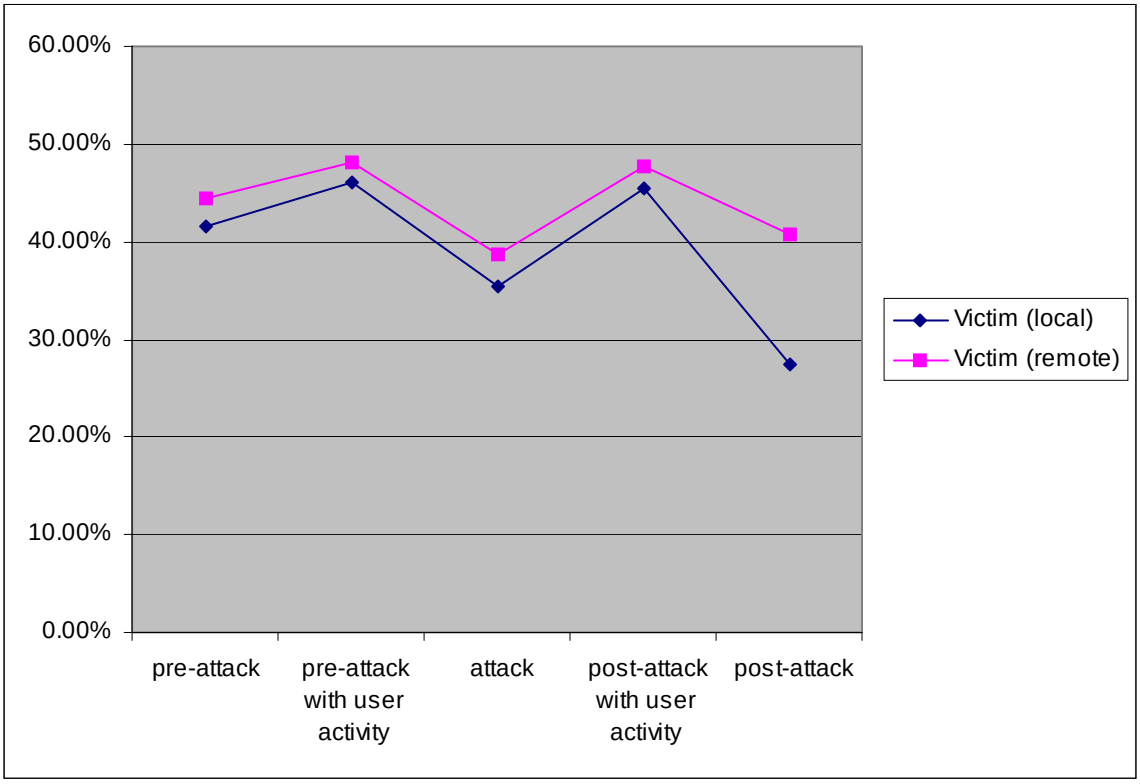


Figure 56. Correlation percentage for both local and remote data collection modes

Again, after screening only the common significant variables and then calculating the Pearson correlation percentages, the difference between local and remote data collection is within a few percentage points (with the exception of the post-attack phase this time).

For the FTP experiment, we also studied the changes in correlation percentages when the same performance variables are examined across all phases of the experiment. As we did for the text editing experiment, we screened the data from all phases of the experiment in both local and remote data collection modes for only the variables that were always non-zero and non-invaried. For the FTP experiment on the victim machine, this was 226 variables, yielding 25,200 correlation pairs. Figure 57 depicts the correlation percentages in both local and remote data collection modes using only the common non-zero, non-invaried variables.

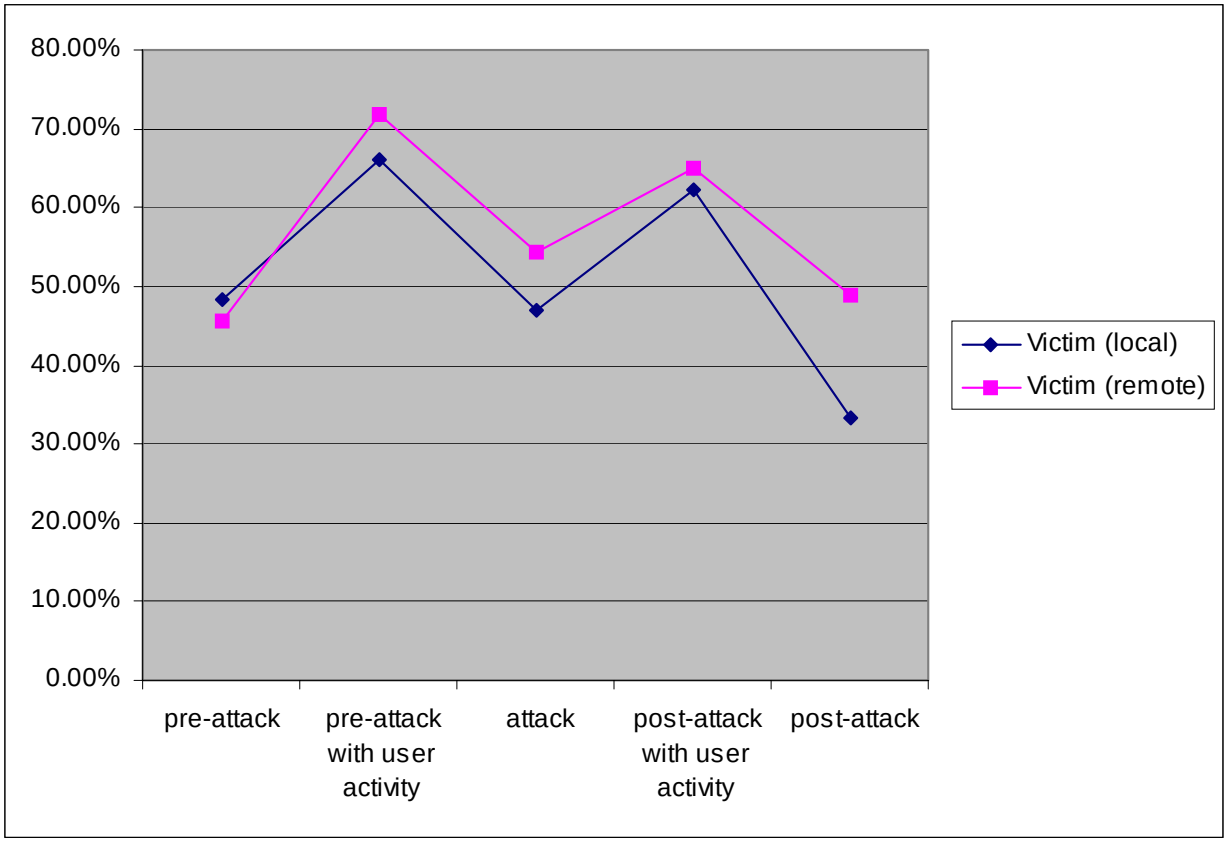


Figure 57. Correlation percentages using only the common non-zero, non-invaried variables

Our observations regarding the characteristics of cyber signal and noise will be presented in the next subsection, independent of the differences in local and remote data collection results.

7.2.3.3 Characteristics of Cyber Signal and Noise from Experimentation with Attack and Normal Data

The Pearson correlation results on a victim machine with normal and attack activity demonstrate the following characteristics of cyber signal:

- The percentage of pairwise correlated variables is lowest during the attack phase (with user activity continuing concurrently) in the FTP experiment. In the text editing experiment, the attack phase has a lower correlation percentage than in the pre-attack phase. These results are similar to the results on the victim machine in the experiment with only attack data. The characteristic of cyber signal that is confirmed is that attack activity lowers the percentage of correlated performance variables in the case of the Sobig e-mail virus, for variables that are non-zero and non-invaried within any given phase. However, if we are to use a set of variables to detect phase transitions, then the results depicted graphically on page 151 must be taken into

consideration. Once these results are taken into consideration, then the most important characteristic of cyber signal that is confirmed is that the attack phase increases the number of non-zero, non-invaried variables.

- In the text editing experiment, the post-attack phase has the smallest correlation percentage of any of the five phases. For the FTP experiment, it has the least number of significant correlations.
- In the FTP experiment, when we examine only the common non-zero, non-invaried variables across all phases for pairwise correlation, the correlation percentage results mirror the results of using different variables for each phase. The correlation percentages were calculated with comparisons between two sets of variables: (1) all of the non-zero, non-invaried variables in each given phase – even though the set of non-zero, non-invaried variables differed between phases – and (2) only the common non-zero, non-invaried variables that had non-zero, non-invaried behavior in every phase. In the experiment with FTP user activity, these correlation percentages for the two sets of variables roughly corresponded to each other. This is useful because in practice one will not know which "phase" of an attack is currently occurring, therefore the results using only the common non-zero non-invaried variables for comparison are potentially useful. This is an encouraging result, in that correlation percentage could be a meaningful indicator of attack phase for the Sobig attack with FTP user activity. To find indications of virus/worm activity with no a priori signature for this virus is an encouraging result, even if experimentation begins in simplistic contexts. However, results for the text editing experiment were different.

The Pearson correlation results on a victim machine with normal and attack activity demonstrate the following characteristics of cyber noise:

- As in the experiments with no user activity, there is a baseline of pairwise correlation between variables, even when there is no user or attack activity, as demonstrated by the pre-attack phase on each machine. This is a form of cyber noise with respect to the performance variable observable point.
- With the start of user activity, in the text editing experiment there is a decrease in the correlation percentage from the pre-attack stage. In the FTP experiment though, there is a slight increase in correlation percentage when user activity starts. This is likely caused by the difference in activity type and its effect on performance variables – highly network-intensive user activity appears to be more correlated.
- The post-attack with user activity phase in the FTP experiment exhibits a similar correlation percentage to the pre-attack with user activity phase, after dipping during the attack phase. This seems to demonstrate again that the form of cyber noise generated by network-intensive user activity is relatively high when compared to other phases of the experiment, and the cyber noise is unaffected by the after effects of the Sobig e-mail virus.
- The post-attack with user activity phase in the text editing experiment has the biggest difference in correlation percentage between the local and remote data collection modes, but when averaged it is slightly lower than the correlation percentage from the pre-attack with user activity phase. Both phases have a lower correlation percentage

than the pre-attack phase (with no user or attack activity), indicating that the cyber noise generated by text-editing user activity seems to lower the percentage of correlated performance variables.

7.3 Task 3 – Investigate, develop and test sensor models of signal detection, and a sensor fusion model for each observable point.

Symantec investigated, developed, and tested sensor models and a sensor fusion model for the database domain using the query time series. The query time series is observable at the database, at the client machine, and on the network in between. In this manner, the query time series is consistent in our experiments at each observable point, and the sensor and sensor fusion models for any one of these observable points then are sensor and sensor fusion models for each of the other observable points in our experiment.

The sensor models are stochastic models created by processing the query time series, either adaptively in real-time or during a training period, characterizing the behavior of either one user or a group of users during a client session. A sequence of queries is modeled as a sequence of "states"; the behavioral model is then a "finite state model" utilizing n-grams.

The sensor fusion models utilize Bayesian networks, in which a probability of attack is calculated from sensor data and prior probabilities.

Testing of the sensors and sensor fusion models occurred via experimentation with database simulations covering a wide range of representative normal usage and attacks. In total, eleven usage and attack scenarios were tested to evaluate the feasibility and correctness of the approach.

The first subsection presents the sensor model developed for the database investigation. The second subsection describes the sensor fusion models investigated and developed. The third subsection presents the testing conducted using simulated database scenarios.

7.3.1 Sensor Model

The key observable point in the database investigation is the processing component of a database for the query time series. The query time series is the sequence of database queries submitted for processing. Because the data collected in the database's query time series is textual (raw SQL query text), it must undergo significant transformations to enable attack detection.

The approach taken in the database investigation was to factor this transformation into several steps, including development of both sensors and a sensor fusion model. The first of these steps is the derivation of both categorical and arithmetic variables from the query time series text and other information. This derivation is done by processing components that "sense" certain features of the data; in accordance with the signal detection approach, these processing components are called "sensors". The choice of sensors is driven by hypotheses developed by investigating and simulating attack scenarios.

The sensors are, in effect, mathematical functions. These functions range from simple functions of single queries to history-dependent functions of the query time series and related query information (such as user name, session ID, IP address, time, etc.).

Relatively simple computations include:

- Query text, with literals, table references, and column references replaced by variables
- Query complexity according to several measures
- Count of number of certain constructs (SELECT, WHERE, JOIN, *, etc.) in query
- Tables/columns referenced by queries
- Sets of bindings for variables, or literals, included in queries
- Measure of aggregation of query content from query to query and measure of breadth of content exploration, observed in sequential dependencies between queries (for a given user or session)
- Inclusion of certain relational operators in queries
- Membership in equivalence classes created by clustering sets of tables, columns, or literal values
- Metrics of breadth of tables and columns accessed and breadth of query constructs employed

More sophisticated computations involve history dependencies. In effect, these computations compute the deviation of query sequences from empirically established norms – deviations with respect to specific characteristics such as those above. The established norms are represented by stochastic models that may be created adaptively in real-time or during a training period.

A discussion of the differences between this investigation and traditional anomaly detection is provided at the end of this section – in particular, our detection approach takes into account both specific attack scenarios as well as deviations from normal behavior, and additional information that supports an attack hypothesis is required and utilized to detect insider attacks. Existing anomaly detection was an appropriate starting point for signal/noise separation analysis in the case of insider attacks against databases, because insider attacks cannot typically be classified a priori as malicious, requiring the integration of adaptive methods to learn legitimate behavior (noise) with attack-specific hypotheses characterizing malicious behavior (signal).

These models characterize the behavior of either one user or a group of users during a client session. A sequence of queries is modeled as a sequence of "states" (equivalence classes of queries). These states are equivalence classes of queries where the equivalence classes are created from the query text by replacing literals, table references, and column references with variables. The behavioral model is then a "finite state model" – specifically, an n-gram model capturing:

- Probabilities of transitions between states, given the *n-1* most recent states
- delays (in time), and variances of delays, between queries (i.e., given a prior sequence of states, the FSM yields the expected value for the delay until the next query, as well as the variance of that delay)
- proportion of visits to each state (separate from transition information)
- relationships between IP and user, as well as IP and query categorization

Other history-dependent quantities are computed separately, such as:

- the length of sessions, measured in time or as a number of queries
- the number of concurrent sessions for a given DB user

With each of these computed as implied above, deviation from historical norms is easily detected for many of them. These signal detection models are similar to those utilized in standard anomaly detection, however:

- The deviations are computed for very specific quantities that have been chosen by considering specific *attack* scenarios, not just normal behavior.
- The computed result is not only a metric of the magnitude of the deviation, but may include other "directional" information; for example, if a sequence of queries occurs at an unusual rate, or with unusual regularity, the directionality of the difference (for both rate and regularity) is recorded – only certain types of changes (e.g., corresponding to a transition from human analysis to automated data collection) correspond to attacks.
- The computed deviation is typically not sufficient, independently, to identify an insider attack. Additional information that specifically supports an attack hypothesis (captured by other variables) is typically required.

Testing of these sensor models is described later in this section.

7.3.2 Sensor Fusion Model

Experimentation with the various transformations above, as well as the study of attack scenarios, strongly suggested that fusion of information was critical to creating reliable indications and warnings.

Symantec first investigated several methods of sensor fusion, including the following:

- **Naïve Bayesian Inference:** A collection of variables are modeled with large joint probability distribution. The advantages of this fusion model are that it is well grounded and conceptually simple. However, it is very difficult to compute and manipulate for a variety of reasons (for example, conditional independence is not assumed).
- **Bayesian Networks [36]:** Specific assumptions of conditional independence are made, leading to a directed graph with nodes representing random variables and edges representing conditional probabilities. Bayesian networks are also well grounded and conceptually simple, and there are optimized implementations available. The requisite assumptions of conditional independence are a disadvantage.
- **Dempster-Shafer Evidence Theory [37,38]:** In some formulations, this is a generalization of Bayesian formalism to handle uncertainty, typically rule-based or using something like random variables. This method does not necessarily require conditional independence. Unfortunately, in practice it is conceptually difficult, computationally cumbersome, and difficult to apply correctly.
- **Certainty Factors [39]:** This is a rule-based formalism, where individual rules have associated uncertainty. This fusion model is simple to understand and implement, but again it requires a conditional independence that is difficult to achieve. In addition, the decision theory is ad hoc.

- **Weighted Average / Voting / Neuron:** A weighted average of variables is computed and compared against a threshold. This model is also simple to understand and implement, but it is not well grounded and loses information about probabilities.

Of these approaches, the Bayesian network fusion model was chosen and implemented. After weighing the advantages and disadvantages of each method, Bayesian networks was determined to be the most amenable to the sensor fusion of database query time series sensor values. Our approach utilizes a Bayesian network model and then infers a probability of attack from various evidence and prior probabilities. Both the inferred variable (in this case probability of attack) and the choice of evidence are configurable. Figure 58 shows an example Bayesian network fusion model derived from experimentation with the scenarios described later in this section.

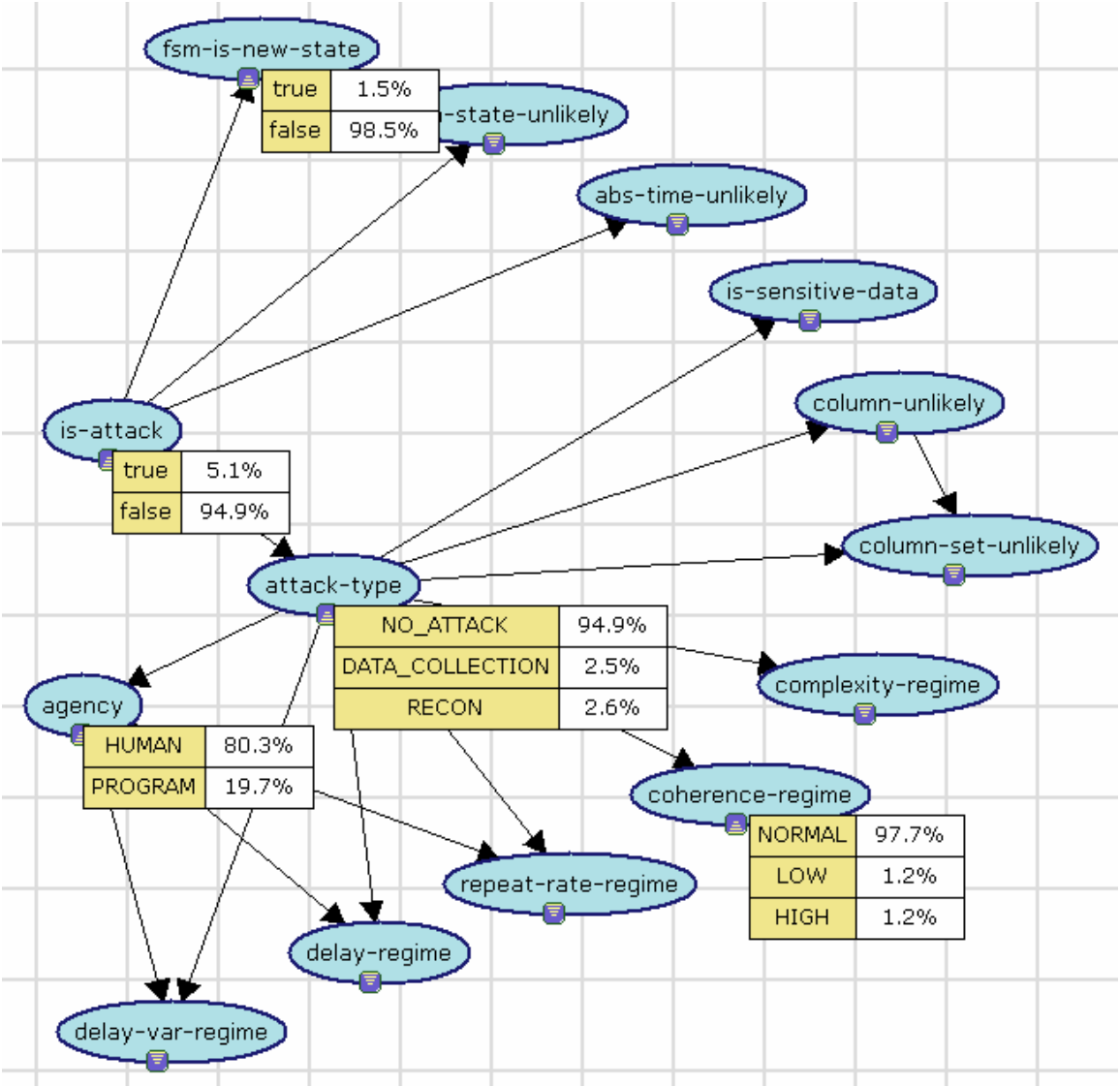


Figure 58. An example Bayesian network fusion model

Testing of this sensor fusion model is described in the next subsection.

7.3.3 Testing of Sensor and Sensor Fusion Models

Testing of both the sensor and sensor fusion models was conducted via experimentation with simulated database scenarios, involving both normal user activity and attack activity. Symantec devised eleven database scenarios across three representative database styles for testing purposes. Testing was conducted primarily to evaluate the feasibility of the approach and the correctness of the implemented sensor and sensor fusion models. An overview of the experimentation conducted for testing will be provided in this subsection.

The database scenarios are classified by the database style that they explore. Modern database management systems have numerous features, only some of which may be relevant to a given database installation. Often the particular features used, and the way they are used, fits a well-known pattern or "database style". The database scenarios investigated for testing considers three database styles:

- Online transaction processing (OLTP)
- Data warehouse
- General purpose (e.g., Web-Note: Some of the scenarios investigated may be more relevant to the IC environment than others; for example, we believe that the data warehouse scenarios would have more relevance than the OLTP scenarios.)

Online transaction processing databases are typically utilized for commerce, inventory, supply chain management, etc. Queries are often issued via stored procedures and are highly stereotyped. Individual queries typically involve small data sets and the rate of queries is very high. Transactions often involve both reading and writing.

Data warehouse databases are typically utilized for decision support, marketing research, analysis, etc. The data is often "not-quite-real-time", but there can be very large data dictionaries, with many relations. Usage of the database involves variable query patterns, including possibly ad hoc queries. Data processed in individual queries may be large.

General-purpose databases are utilized for Web servers and e-commerce. There are multiple classes of users, some sharing a single user identity and others with higher privileges (e.g., to perform administrative operations). The database may sit behind an application server or web server and may support a heterogeneous set of applications. In practice, the features include a mixture from both OLTP and data warehouse databases.

Scenarios are presented by database style. Each database style has its own "story" that provides a means of discussing scenarios. The story is included for pedagogical and motivational reasons, but it is the implementation that determines the breadth of applicability of the scenario. For each database style, an introduction to that style is provided, followed by a listing of the individual scenarios for that database style.

7.3.3.1 OLTP Scenarios

The OLTP scenarios are described with respect to a database of simulated credit card information, including tables of basic account information, payment history, transaction history for a set of cardholders, and case histories of customer service. The simulated database user is a customer service representative (CSR) operating in a highly constrained mode.

Under normal circumstances, the user has no privileges on the data itself, only privileges to execute a particular set of stored procedures, each representing a transaction for part of his/her job function. The stored procedures carry the privileges on the data, and the user's normal functions are carried out through forms-based applications that call the stored procedures. The sensitive data is the basic account information, which includes the cardholder's authentication data as well as the credit card numbers. Attacks are aimed at theft of sensitive data, using the stored procedures available to the user.

Normal activity is organized around customer calls handled by CSRs. Handling a customer call entails performing a sequence of transactions from an allowed set, all carrying the same card number. The malicious CSR interleaves either reconnaissance or data collection activities with his/her normal activities.

The OLTP scenarios are:

OLTP1: CSR retrieves credit card information (manually) without a corresponding customer call

OLTP2: Upon a customer call, the CSR deviates from usual script to retrieve private information about another customer

OLTP3: CSR performs manual reconnaissance

OLTP4: CSR performs automated data collection

7.3.3.2 Data Warehouse Scenarios

The important characteristics of the data warehouse scenarios are:

- the universe of queries is not bounded in normal use
- a single analyst may produce new queries with different structure, accessing different tables, joining them different ways, every day
- a single analyst may execute queries repetitively or only once
- in general, established anomaly detection techniques are less applicable to attack scenarios for this database style

Unlike the OLTP case, where queries and query sequences are highly stereotyped, in the case of the data warehouse, for almost any behavior, there is some database installation for which that behavior is legitimate. It is much more difficult to attach significance, a priori, to individual behaviors. In this case, more sophisticated classification schemes must be employed and deviations from normal behavior can be expected to be more subtle. The opportunity arises because, *for a given installation or user*, only some behaviors will typically be "normal", and the scenarios seem to suggest that certain kinds of deviations from that "normal" will be particularly suggestive of an attack or its precursors.

Two kinds of data warehouse installations are considered:

- A commercial data warehouse used for decision support and marketing analysis. The simulated user is a marketing analyst with broad query privileges.
- A highly secure data warehouse at an intelligence agency, used for analysis of intelligence data. The simulated user is an intelligence analyst with broad query privileges, limited by need-to-know restrictions to particular subject areas.

The specific scenarios are believed to apply, to some extent, to both kinds of data warehouses. Scenarios have been constructed with different levels of access to reflect possible differences in the amount of data available to a single user.

In each case, the malicious user is a highly trusted employee, with privileges sufficient for the user to follow his/her instincts in making new connections between seemingly unrelated data items. Users are able to issue *ad-hoc* complex queries.

Normal analyst activity, in both the marketing and intelligence data warehouse installations, includes a mixture of repetitive and exploratory query sequences. The more difficult problem involves the exploratory sequences. These may illustrate several typical characteristics:

- complex queries may be built up in stages, as the user examines the intermediate results and finds ways to filter and refine them by joining them with information in other tables—the connections the user is trying to establish or refute may involve multiple tables among which there are semantic relationships (usually via foreign keys)
- graphs of relationships might be traced out by self joins on certain tables (e.g. on a table of telephone call records, when A calls B and B has at some point called C, there is a possible connection between A and C) — some sets of queries might be issued several times with different parameters for what-if analyses — a higher frequency of erroneous queries (than found in non-analyst activities)

The analysts have privileges to access database tables directly. They are more likely (relative to the other classes of scenarios) to examine the database structure to formulate ad-hoc queries linking several tables; thus, metadata queries have a significantly lower chance of signaling the onset of an attack. There is a wider range of activities between highly focused retrievals and systematic extraction of larger volumes of data.

The data warehouse scenarios are:

DW1: Attacker performs reconnaissance, manually exploring breadth of data
DW2: Collecting data, attacker does not form query by accreting terms in query
DW3: Attack involves transition from exploratory to repetitive data collection
DW4: Attack involves topic outside historical norm for user
DW5: Attack performed at time outside historical norm for user

7.3.3.3 General Purpose Scenarios

The general-purpose database (as might be found behind a Web server, portal, or application server) supports some transaction processing as well as some analysis functions (for users as well as administrators).

Because of the complex functionality of Web servers, they often have significant configuration vulnerabilities. When they are penetrated, the attacker often gains the same rights to the database as the Web server itself. Thus, this is one case where misuse detection provides a last line of defense against other types of attacks.

Normal usage comprises the Web-based transactions and some server administration. Conceptually, this includes user login/logout, access to public and private areas, searches for and display of content, financial transactions, and access of private content by an administrator.

We examine two attack scenarios. The first deals with the case where the Web server has been taken over. In this case, the attacker uses the database user ID normally used by the Web server and retrieves sensitive data. The second deals with the well-known but persistent problem of SQL injection.

The general-purpose database scenarios are:

GP1: Inappropriate use of web server user privileges to access otherwise unavailable data

GP2: SQL injection to get otherwise private data

7.3.3.4 Testing Results and Observations

Each of the eleven scenarios described above was implemented and tested using an augmented and instrumented Oracle database infrastructure. In each scenario run, both normal user activity and attack activity were simulated, and in each scenario the sensor models and sensor fusion models were calculated over the query time series. In the majority of the scenarios, the attack queries were detected after a training period, albeit with varying degrees of false positive performance. For the remaining scenarios, further analysis is required to determine the cause of missed detections (e.g., insufficient training, inadequate models, etc.). These initial testing results demonstrated the feasibility of the approach (with simulated data) and implementation; further investigation and evaluation is required to fully validate the utility of the approach.

Experimentation with the Bayesian network approach to sensor fusion suggests that it is viable; however, many of the conditional probabilities must be adjusted to get a desired frequency of alerts and a relative weighting of evidence that reflects the certainty about the relevance of the evidence – for some scenarios some evidence is much more useful than others. For example, in some attack scenarios, the sensor for whether the data accessed by a query is sensitive (called “is-sensitive-data”) is a very important indicator: in that case, for optimum effect (to correctly identify the attacks) the correlation between “is-sensitive-data” and the attack determination (“is-attack”) can be artificially increased relative to other variables that are deemed less relevant. If not careful, this artificial increase can be exaggerated to a point not accurately reflecting risks and sensitivity of aggregation.

7.4 Task 4 – Formulate and solve an optimization problem to select an optimized suite of I&W observables / cyber sensors.

Symantec formulated and solved an optimization problem to select an optimized suite of I&W observables / cyber sensors.

The optimization problem minimized the "harm" in terms of whether or not to alert an administrator of a potential attack. The I&W observable that comprised the optimized suite was the database query time series. Experimentation with the database scenarios yielded some useful cyber sensors for the optimized suite, such as a new state being added to the finite state machine, irregularly timed queries, unusually complex queries, and unusual use of columns.

The first subsection presents the formulation of an optimization problem for I&W in the database domain and the solution utilized for this investigation. The second subsection discusses the optimized suite of I&W observables / cyber sensors.

7.4.1 Optimization Problem for I&W of Insider Database Attacks

In formulating the optimization problem for providing I&W of insider database attacks, several components of an I&W system were considered. These include the database and its users, human security administrators, and the I&W infrastructure itself.

7.4.1.1 The Database and Its Users

Database queries are associated, by the database, with database users (DB users). A given DB user may issue queries on behalf of more than one human or organization, for example, in circumstances where a single DB user account is shared between a web server and the database. For the purposes of our research addressing insider attacks, it is assumed that each query can be associated with a single DB user that is responsible for initiating it. The approach presented here does not depend on the specifics of that DB user, but a required property of a DB user is that it may be ascribed malicious intent or legitimate intent for each query that it initiates.

Several pieces of information are almost always observable for each query, including the time of the query,

$$t = \textit{time of the query},$$

as well as the query content itself (i.e., the SQL statement text), the DB user issuing the query, the IP of the host that sent the query to the database, and other information,

$$q = (\textit{query SQL, DB user, IP of host, ...}).$$

Because each query can be identified with a DB user who can in turn be ascribed malicious or legitimate intent, queries can be organized into "incidents". For the purposes of this discussion an incident can be defined:

- by defining "sub-incident" as a contiguous sequence of queries issued by a single DB user with malicious intent; and
- by defining "incident" as a sub-incident that is not part of any other sub-incident.

Thus, given a time series of queries, (q_0, t_0), ..., (q_i, t_i), the incidents can be numbered (1, 2, ...). Then, a given query, q_i, can be associated with a given incident number (or, say, zero if the query is not part of an incident), a_i.

In practice, a query (or collection of queries) must meet two tests in order to be determined to be part of an incident:

- the query (or collection of queries) is statistically anomalous, where
- the specific character of the anomaly is consistent with some prior "hypothesis" about how an attack might take place.

While the domain-specific problem of defining an incident is important for a production system – it may, for example, impact how evidence of an attack is aggregated and presented to an administrator – the database investigation focused on the question of whether an individual query was part of an incident.

7.4.1.2 Human Security Administrators

When an I&W system "detects" an incident, a human security administrator (HSA) will typically be notified (perhaps in addition to other things occurring). In the case of insider database attacks, this contributes a significant amount of complexity to the problem of developing optimized I&W: the costs and benefits of notifying an administrator are difficult to quantify.

The cost of notifying an administrator will depend on what is expected of him or her. At the very least, some time is consumed validating the detection. How much time will depend on what type of forensic information is provided. Since many possibilities may contribute to a decision to notify the administrator – including mutually exclusive possibilities – there is a great deal of variety in the type of information that presumably should be provided. For example, a system might:

- identify a single query
- identify a DB user
- identify an incident
- identify several queries, DB users, or incidents
- merely indicate that "something is wrong somewhere"

Quantifying the cost is further complicated by the fact that the cost of a false positive depends on what action the administrator takes (e.g., shutting down the database).

The benefits of notifying an administrator, aside from potentially preventing or mitigating damage from an attack, include the possibility of training the I&W system. Since the administrator has pattern recognition skills and "out-of-band" knowledge about the database system and its users, the administrator will generally know things that the I&W system does not. By doing a forensic investigation, the administrator can feed back information (e.g., "this was not an attack"). Further, this information may, in principle, be very specialized (e.g., user, John Doe, is supposed to be doing this).

The present research takes a very simple approach, assuming a fixed expected cost per notification and some fixed expected loss should an attack occur and not be detected.

7.4.1.3 I&W Infrastructure

As noted previously, a defining property of insider attacks is that the behavior that constitutes the attack cannot be classified *a priori* (i.e., before any kind of training) as malicious. This is very different from cases where attacks have some clear signature (e.g., network traffic increased by orders of magnitude in a DOS attack). Given a query time series, the perceived probability that an attack is underway will tend (in many situations) to be small even when an attack is actually under way – i.e., in this situation, the specific choices made in detection and analysis methodologies may more frequently impact the results.

Analysis of the scenarios considered in this research reveals that a significant amount of uncertainty is inherent in detection of insider attacks and that a general approach must account for such uncertainty.

Further, while determining whether an attack is underway is a challenge, determining what to do about it also creates a challenge. Per the discussion above, there are many variables in notifying a human security administrator. Further various actions may be automatically taken:

- block the query (and any transaction it is part of to satisfy integrity constraints)
- block all queries by the DB user
- shut down access to a table
- etc.

The present research considers a passive system in which an attack, or pending attack, is detected and an HSA is notified, but no other action is taken. Even in this case, notification of an HSA has a cost – the cost of the HSA's attention for a period of time.

Yet further, initial research on database attack scenarios suggests that different types of attacks may lend themselves to qualitatively different detection schemes. Thus, it is anticipated that the I&W system will have multiple "hypothesis-specific" modules to explicitly deal with these different schemes. Information from these different modules must be merged. This necessitated a data fusion approach within the database-specific I&W infrastructure, even before integrating with other I&W components.

Because of the importance of uncertainty in detecting insider attacks, it is useful to consider a probabilistic formulation of the I&W problem.

As noted above, determination of the "maliciousness" of a query can only be made in light of historical information (i.e., training, whether it be in a dedicated "training" session or through observing a production system). Such historical information can be captured in stochastic model.

Consider the time series discussed above,

$$(q_0, t_0, a_0), (q_1, t_1, a_1), \ldots, (q_j, t_j, a_j) \; ; \; t_j > t_{i+1} > t_i > t_0$$

to be an outcome of the stochastic process,

$$X = \{X_i : i \in N\} = (Q, T, A) \; ; \; X_i = (Q_i, T_i, A_i),$$

where X_i is a random variable with the sample space,

$$\text{"queries"} \times \text{"time"} \times \text{"incident number"}.$$

The process X represents a model of the "real world", including information about whether various queries were parts of incidents. Then, of course, a simple determination of whether to warn a human security administrator (HSA) might then depend on whether

$(P_a * h(\text{“notified”, “attack”}) + (1–P_a) * h(\text{“notified”, “no attack”})) <$
$(P_a * h(\text{“didn’t notify”, “attack”}) + (1–P_a) * h(\text{“didn’t notify”, “no attack”}))$,

where,

$h(\text{“whether notified”, “whether attack was real”}) = \text{“metric of harm”}$,

and P_a represents the perceived probability that an incident is an attack,

$P_a = P\ (A_i > 0\ |\ Q_0 = q_0, T_0 = t_0, ..., Q_i = q_i, T_i = t_i)$.

Note that metric of “harm”, *h*, like the process, *X*, may reflect historical information (e.g., *h* may reflect the fact that it may be more expensive to notify an HSA more frequently than some optimal rate).

In the present research, we used the following model where the harm H_i for each incident *i* could have any of three possible values:

- h(“didn’t notify”, “no attack”) = 0,
- h(“notified”, “attack”) = c
  = h(“notified”, “no-attack”) = “cost of notification” (or “cost-per-alert”), and
- h(“didn’t notify”, “attack”) = ℓ = “prospective loss from attack” (or “prospective-loss”).

As described above, we assume a fixed expected cost per notification even if there was no attack, a fixed expected loss (assumed to be a greater harm) for attacks without notification, and no harm (other than the cost of the alert) for attacks with notifications. The decision function from above

$(P_a * h(\text{“notified”, “attack”}) + (1–P_a) * h(\text{“notified”, “no attack”})) <$
$(P_a * h(\text{“didn’t notify”, “attack”}) + (1–P_a) * h(\text{“didn’t notify”, “no attack”}))$,

may then be simplified to read:

$$[P_a * c + (1 - P_a)*c] < [P_a * \ell]$$

which reduces to:

$$c < P_a * \ell$$

and further reduces to:

$$P_a > c / \ell$$

In other words, the decision logic might be stated, “warn an HSA when the perceived probability of an incident being an attack is greater than the cost of a false alarm divided by the potential loss of a missed attack.”

In this optimization problem, the goal is minimization of total harm *H* as a sum of all harms H_i for all incidents in a time series sequence of incidents:

$$H = \sum_{i=0}^{j} H_i$$

However, as noted above, the magnitude of each harm is a function of both ground truth of whether or not there was actually an attack in progress, as well as the result of the decision function as to whether or not an alert went to an HSA. For this reason, at this point it is necessary to more formally represent ground truth. We represent ground truth (*G*) of each incident as either actually being an attack ($G_i=1$) or actually not being an attack ($G_i=0$). Given this representation of ground truth, and decision logic described above, the four cases of the truth table may then be represented as:

If $G_i = 0$ *and* $[(c/\ell) - P_a] >= 0$ *then* $H_i = 0$

If $G_i = 0$ and $[(c/\ell) - P_a] < 0$ then $H_i = c$

If $G_i = 1$ *and* $[(c/\ell) - P_a] < 0$ *then* $H_i = c$

If $G_i = 1$ *and* $[(c/\ell) - P_a] >= 0$ *then* $H_i = \ell$

This may be represented in closed form as

$$H_i = [\, c * [\, POS\, (\, P_a - (\, c / \ell\,)\,]\, * (\, P_a - (c / \ell\,)\,)\,] + [\ell * G_i * POS\, (\, (\, c / \ell\,) - P_a\,)\,]$$

where *POS (x)* is a function such that:

POS (x) = 1 for x > 0,
POS (x) = 1 for x = 0, and
POS (x) = 0 for x < 0.

Of the first term of the closed-form representation of H_i, the first component, namely

$$c * [POS\, (\, P_a - (\, c / \ell\,)\,]$$

captures the two cases where alerts are generated and the harm is equal to the costs of the alerts.

Of the first term of the closed-form representation of H_i, the second component, namely

$$(P_a - (\, c / \ell\,)$$

captures the boundary case where the perceived probability of an incident being an attack precisely equals the cost of an alert divided by the loss of attacks without warning. In such cases, in the decision logic above, an alert is not generated. In such cases, the harm is entirely a function of ground truth G_i.

The second term of the closed from representation of H_i, namely

$$[\ell * G_i * POS\, (\, (\, c / \ell\,) - P_a\,)\,]$$

captures the case where an alert is not generated though an attack is imminent or occurring. In such cases, the harm is the loss from an attack without warning.

Given then that c and ℓ represent constants, most likely derived from historical or projected costs and losses, and given that indications and warnings systems have no control over ground truth G_i, indications and warnings systems then only have one means to minimize the harm H of a time series sequence of incidents. That means is the means of adjusting the accuracy of P_a in estimating the likelihood of an attack. The optimization problem then is minimizing H by improving accuracy of P_a. To do this, we must more formally define the accuracy of P_a. To do this, we revisit H_i:

$$H_i = [\, c * [\, POS\, (\, P_a - (\, c / \ell\,)\,]\, * (\, P_a - (c / \ell\,)\,)\,] + [\ell * G_i * POS\, (\, (\, c / \ell\,) - P_a\,)\,]$$

We might simplify this by defining a condition w where the HSA is warned of a potential attack. From the truth table we note that

$$H_i = [\, c * w\,] + [\ell * G_i * not\ (w)\,]$$

where *not (x)* is the boolean function "not."

Without loss of precision, this may be rewritten as:

$$H_i = [\, c * w * G_i \,] + [\, c * w * not\,(G_i)\,] + [\ell * G_i * not\,(w)\,]$$

And further re-written as:

Equation Z:

$$H_i = G_i * [\, w * c + not\,(w) * \ell\,] + [\, not\,(G_i) * w * c\,]$$

Given the assumption that the average cost c of each alert is less than the average loss ℓ of each unwarned attack,

$$[\, c * w * G_i\,] < [\ell * G_i * not\,(w)\,]$$

Therefore, to minimize the first term Equation Z, namely to minimize the quantity $G_i * [\, w * c + not\,(w) * \ell\,]$, it is necessary to maximize correspondence of w with G_i.

Examining the second term of *Equation Z*, we note that minimizing the second term of *Equation Z*, namely minimizing the quantity $[\, not\,(G_i) * w * c\,]$ requires minimizing correspondence of $[\, not\,(G_i)\,]$ with w.

In other words, given that *Equation Z* represents harm, minimizing harm requires maximizing correspondence of w with G_i while minimizing correspondence of w with $[\, not\,(G_i)\,]$. "Optimal" minimization of harm then would be cost c for each G_i with no penalties ℓ for G_i with no w, and no penalties c for w with no G_i. For this reason, optimization in minimizing harm has a linear relationship with minimizing quantity M where:

$$M = [\, c * w * not\,(G_i)\,] + [\ell * G_i * not\,(w)\,]$$

Given that c and ℓ are constants, this is effectively similar to minimizing M' where:

$$M' = [\, (c / \ell) * w * not\,(G_i)\,] + [\, G_i * not\,(w)\,]$$

We note from the decision logic that w is a function of P_a:

$$w = [\, POS\,(P_a - (c / \ell)\,] * [\, P_a - (c / \ell)\,]$$

We recall the definition of Pa as

$$P_a = P\,(A_i > 0 \mid Q_0 = q_0, T_0 = t_0, \ldots, Q_i = q_i, T_i = t_i).$$

We then consider P_a as a function of the histories of observables in Q, T, and A. Optimizing the indications and warning system then is the challenge of minimizing harm by selecting sets of observables in Q, T, and A, and selecting functions for P_a such that $P_a = f\,(Q, T, A)$ generates a sequence of warnings w_i that minimize M'.

Given this then, the solution of this optimization problem then lies in selecting the suite of observables, sensors, and functions that generate sequences of warnings that minimize M'.

The I&W observable that comprised the optimized suite was the database query time series. Experimentation with the database scenarios yielded some useful cyber sensors for the optimized suite, such as a new state being added to the finite state machine, irregularly timed queries, unusually complex queries, and unusual use of columns.

7.4.2 Optimized Suite of I&W Observables / Cyber Sensors

Given our formulation of the optimization problem for I&W of insider database attacks, data from the sensors described in the previous section could be utilized for determining whether or not to alert on detection of a possible attack.

In order to facilitate testing with the database scenarios mentioned in the previous section, Symantec developed a demonstration and evaluation prototype, implementing the database sensors observing the database query time series. The Symantec database prototype is configurable with the "cost of notification" and the "prospective loss from attack" values that enable solving the optimization problem (again, minimizing the harm in terms of whether or not to alert an HSA on a prospective attack). For each query in the query time series of a given simulated scenario, the probability of attack is calculated and the prospective benefit (i.e., reduction of harm) of alerting an HSA is calculated to determine whether or not to alert on the query.

For the optimized suite of I&W observables / cyber sensors, each observable value is calculated for each query in the query time series – the transformations required on the query time series to calculate each observable value in general do not preclude their calculation for each query, in each database scenario. However, some observables / sensors demonstrated more utility than others in terms of calculating the probability of attack, depending on the database scenario being tested.

The following sensor types demonstrated utility in calculating the probability of attack during experimentation with the database scenarios and attacks:

1. fsm-is-new-state: a sequence of query types occurs that has not previously been seen (OLTP1, OLTP4, and GP1 scenarios)
2. delay-regime: slow queries, indicative of reconnaissance activity (OLTP3, DW4, and GP1 scenarios)
3. delay-var-regime: irregularly timed queries, again indicative of reconnaissance activity (OLTP3, DW4, and GP1 scenarios)
4. complexity-regime: unusually complex query, indicative of reconnaissance (OLTP3 and DW4 scenarios)
5. coherence-regime: unusual lack of variation in queries (OLTP3 scenario)
6. column-unlikely: a rarely seen column (OLTP3, DW4, and GP1 scenarios)
7. column-set-unlikely: a rarely seen set of columns (DW1 and DW4 scenarios)

Please note that data from these sensors, and possibly others, is fused when making the determination of notifying an administrator of a potential attack – thus, no single sensor variable alone is necessarily indicative of an attack. Please also note that optimization was only done for the following scenarios:

- OLTP1
- OLTP3
- OLTP4
- DW1
- DW4
- GP1

For this set of scenarios, pruning any of the sensors decreased the suite's ability to reliably detect such breadth of attacks. Moreover, for this set of scenarios, adding additional scenarios either introduced additional false-positives, each carrying cost *c* for each additional false positive, or simply did not reduce either false positives or false negatives. In this regard the set of sensors above represents an optimized suite of cyber sensors corresponding to I&W observables.

7.5 Task 5 – Test and verify research outcomes using real information infrastructure data that is available at Symantec.

Symantec tested and verified research outcomes from ASU using real information infrastructure data available at Symantec.

ASU provided research outcomes via technical reports and presentations. Symantec possesses real information infrastructure data, in the form of Sobig e-mail virus samples, which could be run in secure laboratory facilities only within Symantec. These samples consist of code and data obtained from real information infrastructure. Using these samples, Symantec generated additional data to enable testing and verification of research outcomes from ASU.

The research outcomes from ASU took the form of performance variables that were determined to be possible indicators of cyber attack via a variety of statistical analyses. Symantec was able to test and verify a subset of these research outcomes using the Sobig sample and generated data, as outlined below.

The first category of research outcomes provided by ASU related to probability distributions of performance variables. In that section, the common performance object variable groups that shift distribution between phases were Memory, Process, Processor, and Terminal Services Session. In Symantec's testing with data generated from Sobig experimentation, the first three of these performance object variable groups – Memory, Process, and Processor – also shifted distributions. Terminal Services Session could not be verified; this difference may be caused by differences in performance variables provided by Microsoft for different types of systems (e.g., laptops versus desktop machines, Windows 2000 versus Windows XP).

The second category of research outcomes provided by ASU and verified by Symantec related to difference in means (averages) of performance variables. ASU lists 32 example performance variables that shift averages among phases for all six of their investigated attacks. Of these 32 variables, the 24 variables that related to the Memory, Objects, and Process performance objects were also confirmed to shift averages in Symantec's testing with data generated from Sobig experimentation. The other 8 variables that could not be confirmed all related to the Terminal Services Session object. Absence of Terminal Services Session data appears to be a result of differences in performance variables provided by Microsoft for different types of systems.

In sum:

- of the 4 performance object variable groups demonstrating shifts in distributions in evaluation at ASU, 3 of those 4 object groups demonstrated similar shifts in distributions during evaluation at Symantec.

- of the 32 performance variables demonstrating shifts in averages in evaluation at ASU, 24 of those 32 performance variables demonstrated similar shifts in averages during evaluation at Symantec.

In this regard, Symantec verified research outcomes regarding 3 of 4 performance variable object groups and 24 of 32 specific performance variables as generalizing across the different experimentation environments and different attacks between ASU and Symantec. The remaining research outcomes could not be verified as generalizing, most likely due to differences in performance variables provided by Microsoft for different types of systems, or differences in interaction of the operating system with underlying hardware across different hardware platforms. Most specifically, the research outcomes that differed all related to Terminal Services Session object variables, and the software provided to Symantec by Microsoft consistently did not capture Terminal Services Session data. However, we see it as a positive result that 27 of 36 research outcomes (3 of 4 object groups and 24 of 32 performance objects) did generalize across alternative hardware, operating systems, and attack sets, and that all non-verified outcomes might be a result of a single "common cause" of inconsistency of behavior of Microsoft software across versions and hardware platforms. However, the false positive rate is not yet known for production deployment of any of the 27 possible indicators.

7.6 Task 6 – Provide documents that reflect monthly status and final technical report.

Symantec provided documents and e-mails reflecting monthly status. Symantec is also providing this final technical report.

7.7 Task 7 – Participate in project meetings as necessary.

Symantec participated in project meetings as requested every quarter, either in person or via telecon. Symantec also attended all three ARDA PI meetings.

The following is a (partial) listing of quarterly team meetings and ARDA PI meetings attended in person or via telecon by Symantec representatives:

- 11/19/2003 – 11/20/2003: ARDA PI Meeting in Nashville, TN; attended by Juanita Koilpillai
- 2/3/2004 – 2/5/2004: Quarterly status meeting at ASU; attended by Juanita Koilpillai
- 6/6/2004: Quarterly status meeting via telecon; attended by Juanita Koilpillai and Matthew Elder
- 6/22/2004 – 6/24/2004: ARDA PI Meeting in La Jolla, CA; attended by Matthew Elder
- 8/10/2004: ARDA Site visit to ASU; attended by Matthew Elder
- 12/1/2004: Quarterly status meeting at ASU; attended by Matthew Elder
- 1/11/2005 – 1/13/2005: ARDA PI Meeting in Destin, FL; attended by Matthew Elder

7.8 Conclusion

Symantec:

- Collected and examined known cyber attack cases and scenarios to develop threat and attack profiles. These attack cases included database attack cases and virus/worm attack cases.
- Discovered characteristics of cyber signal and noise (attack data and normal data) at each observable point. These characteristics included Pearson correlation percentages for attack data and normal data collected remotely and locally.
- Investigated, developed, and tested sensor models of signal detection, and a sensor fusion model for each observable point. These sensor models are stochastic models created by processing the query time series, and the fusion model investigated, developed, and tested was a Bayesian fusion model.
- Formulated and solved an optimization problem to select an optimized suite of I&W observables / cyber sensors. The optimized suite included delay regimes and unlikely columns utilized.
- Tested and verified research outcomes using real information infrastructure data that is available at Symantec. The real infrastructure data included virus and worm samples taken from real information infrastructure. The outcomes tested and verified included shifts in distribution and averages of the Memory, Objects, Process, and Processor performance objects and variables.
- Provided documents that reflect monthly status and this final technical report. Monthly status documents are included again in Appendix A for convenience.
- Participated in project meetings as necessary, including ARDA PI meetings in November 2003, June 2004, and January 2005, and quarterly status meetings in February 2004, June 2004, August 2004, and December 2004.

Moreover, towards verification of the fundamental hypothesis that a signal and noise detection and separation approach might be useful to cyber indications and warnings – the fundamental hypothesis underlying the ASU research effort and our subcontract – Symantec verified shifts in probability distribution of 3 performance object groups and shifts in averages of 24 performance variables amongst phases of attacks, across platform differences and differences in attacks between ASU and Symantec.

Additionally, in pursuing application of the ASU methodology to other attack classes manifesting themselves in other data streams, Symantec was able to identify 7 cyber sensors as an optimized suite useful in calculating the probability of attack. We consider each of these 7 sensors and verification of each of these 27 performance object and variable research outcomes to each be a substantial finding.

A logical next step, beyond the scope of this contract, might be evaluation of false-positive rates and receiver operator characterization on larger and live production systems for each variable at multiple thresholds of deviation and each sensor along with the sensor fusion model. However, as mentioned previously, and consistent with the direction of ASU, we

consider the 7 sensors and verification of 27 performance object and variables to be substantial findings. Taken in the context of completion of all the tasks mentioned above, these results and this report then fulfill our responsibilities for the technical tasks of the subcontract.

8. AT&T Final Report

The Internet is a global, decentralized network comprised of many smaller interconnected networks. Networks are largely comprised of end systems, referred to as hosts, and intermediate systems, called routers. Information travels through a network on one of many paths, which are selected through a routing process. Routing protocols communicate reachability information (how to locate other hosts and routers) and ultimately perform path selection. A network under the administrative control of a single organization is called an autonomous system (AS). The process of routing within an AS is called *intradomain routing*, and routing between ASes is called *interdomain routing*. The dominant interdomain routing protocol on the Internet is the Border Gateway Protocol (BGP). BGP has been deployed since the commercialization of the Internet, and version 4 of the protocol has been in wide use for over a decade. BGP works well in practice, and its simplicity and resilience have enabled it to play a fundamental role within the global Internet. However, BGP has historically provided few performance or security guarantees.

The limited guarantees provided by BGP often contribute to global instability and outages. While many routing failures have limited impact and scope, others lead to significant and widespread damage. One such failure occurred on 25 April 1997, when a misconfigured router maintained by a small service provider in Virginia injected incorrect routing information into the global Internet and claimed to have optimal connectivity to all Internet destinations. Because such statements were not validated in any way, they were widely accepted. As a result, most Internet traffic was routed to this small ISP. The traffic overwhelmed the misconfigured and intermediate routers, and effectively crippled the Internet for almost two hours.

In this project, we aimed to provide solutions to BGP security through cybersignal detection. Our objective was first to characterize the types of attacks that can be made against BGP and the Internet routing fabric, and then to understand trends in the routing data that characterize BGP traffic and help differentiate attacks and other notable events from random noise in the system. We fulfilled the first seven objectives we faced prior to the end of our contract in September, 2004.

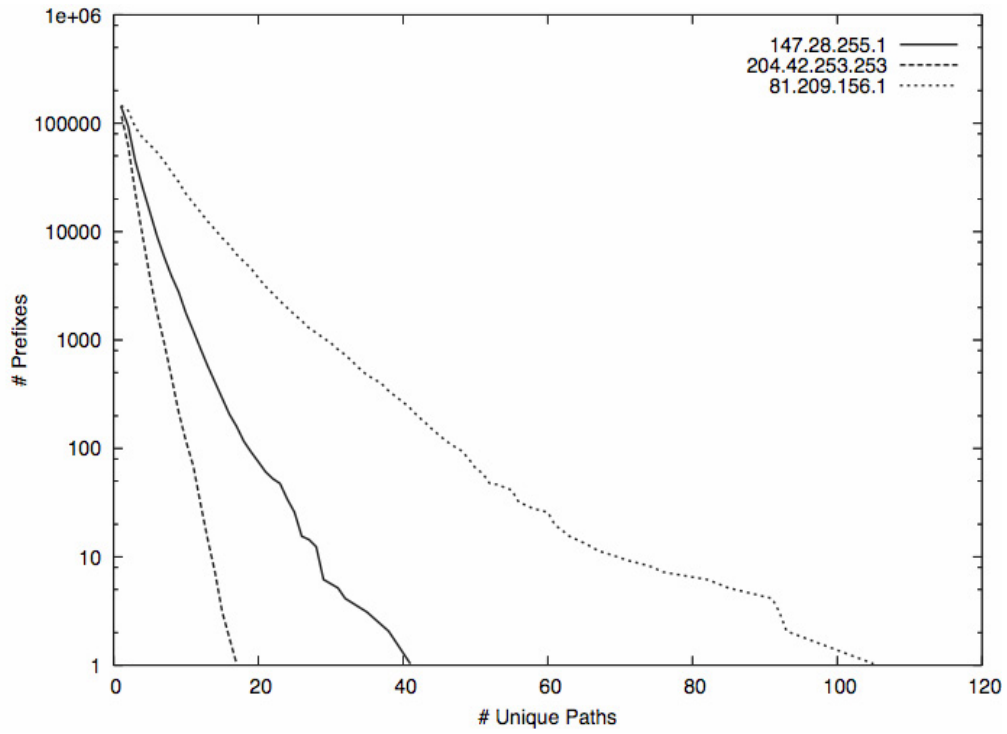


Figure 59. Number of unique paths per prefix. This is an indicator that paths in BGP are very dense, giving a good indication of how to detect signal accordingly.

8.1 Deliverables

We have identified a number of notable characteristics in the global BGP routing data. Using the publicly-accessible RouteViews repository of BGP data from over 40 routers located in autonomous systems around the world, we have created tools to effectively filter the data to expose interesting and important trends and identifying characteristics. Examining over 218 million BGP UPDATE messages, we made discoveries as to the *tail mass* of BGP path vectors, and found the number of unique paths per advertised prefix and autonomous system are generally very dense and stable, as shown in Figure 59. Our inquiries into rate of discovery of new unique paths also uncovered a periodicity to the rate at which paths are added, implying that the global network is most stable on weekends. This is shown in Figure 60.

8.1.1 Collect known attack cases and scenarios.

We have thoroughly investigated the threats and attack scenarios against BGP, and have created a comprehensive list of current countermeasures against attacks. Please see the next section for a full chart showing potential BGP attacks. Additionally, we have outlined a full threat model for BGP, a novel contribution to the research literature. Our work has been summarized in an AT&T technical report awaiting submission as a journal. Examples of attacks against BGP are numerous. Attacks against confidentiality are those where the channel over

which two parties communicate could be subverted by a third party through eavesdropping or other passive attacks, including those against the underlying TCP transport protocol (e.g., SYN flood, RST attack, sequence number guessing). Integrity attacks are those where the attacker does not merely scan the channel, but actively tampers with BGP messages. Message *insertion*, *deletion*, *modification*, and *replay* attacks are possible through these methodologies. Larger scale attacks include fraudulent advertisement of origin information and the subversion of path information. We have examined these attack vectors in great detail and have reported our findings in papers on origin and path authentication in interdomain routing, that have been accepted and submitted, respectively, to major networking and security conferences.

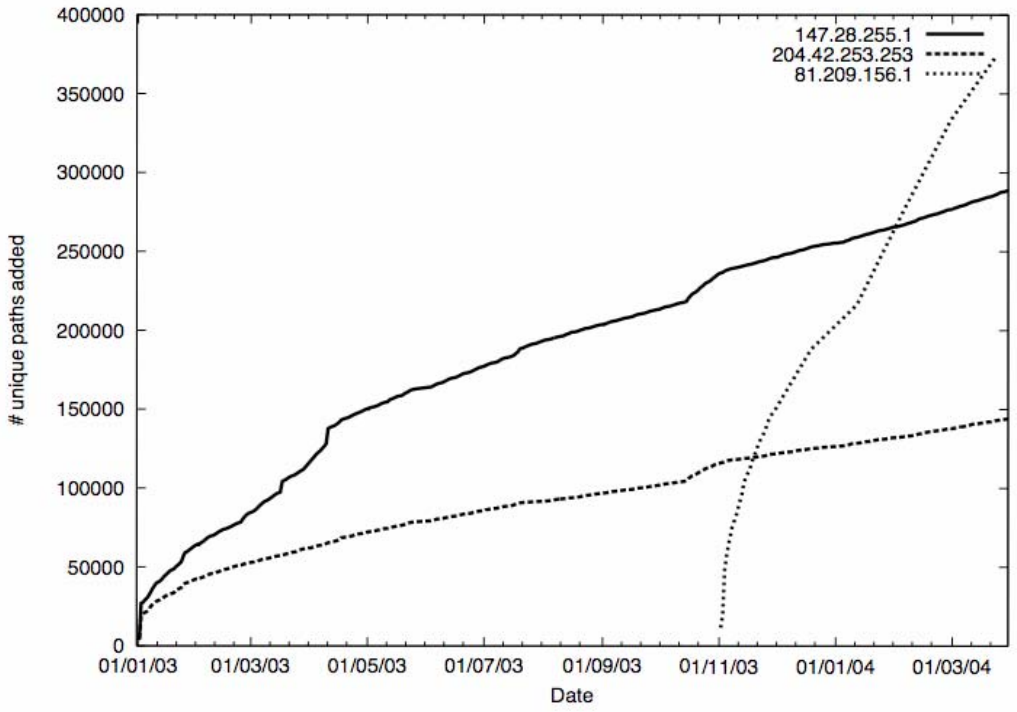


Figure 60. *Rate of Discovery*: the number of new unique paths discovered aggregated by AS. This graph shows that for all listeners, there are relatively few new unique paths added on a daily or weekly basis, another good signaling indicator.

8.1.2 Examine each attack scenario or case to derive the cause-effect network for the attack scenario.

In our technical report, we identify the parties that are liable to initiate attack sequences and examine he effects that these attacks will have within individual autonomous systems, as well as potential effects on the global Internet as a whole. As an example of this, we consider the

denial of service attack on BGP speakers through the RST attack. A remote attacker spoofs a TCP RST message to a router's connection with a BGP peer, causing the router to lose its connection. The resulting effects are greater than the loss of two-party communication: because BGP requires distributed computation, if a router goes offline, then when it comes back online, its routing table will need to be recreated. As a result, it re-announces all of the prefixes it is originating, a process known as a *table reset*. The neighboring routers dump their BGP tables to the peer that has just come online so that it has full data for making its routing decisions. Sifting through this information places a considerable computational burden on the router, and delays processing of normal traffic. If the router is continually knocked offline, the routes it advertises will disappear and reappear in peer routing tables. This is called *route flapping* and is detrimental to all routers, as extra computation and reconfiguration of routes becomes necessary if this happens often. In order to lower the burden, unstable routes are often penalized through a process called *route dampening*. Neighboring routers will ignore advertisements from the router for an increasing amount of time, depending on how often the route flapping occurs. We consider other attacks in our report that follow a similar cause-effect derivation.

8.1.3 Examine the attack scenarios and cases to develop threat profiles.

The comprehensive threat model that we have devised examines the potential attack scenarios and profiles the nature of each threat in order, based on empirical understanding on trends within the BGP data, as well as the results from previous academic contributions to the field and data from leading researchers within the networking community. We consider the ramifications of a dysfunctional routing system under attack. An individual router is subject to being overloaded with information, knocked offline or taken over by an attacker. An autonomous system can have its traffic black-holed or otherwise misrouted, and packets to or from it can be grossly delayed or dropped altogether. Malfunctioning ASes harm their peers by forcing them to recalculate routes and alter their routing tables. We have considered profiles of these attacks in our published and submitted papers.

8.1.4 Develop attack profiles by enlarging the cause-effect network of each attack scenario with threat elements by putting the attack scenario under an applicable threat profile.

The information that we collected and surveyed through the academic literature was collected and displayed as a chart showing a taxonomy of attacks, and further expanded upon in the survey we prepared that catalogued attacks and countermeasures. This table is given in the following section.

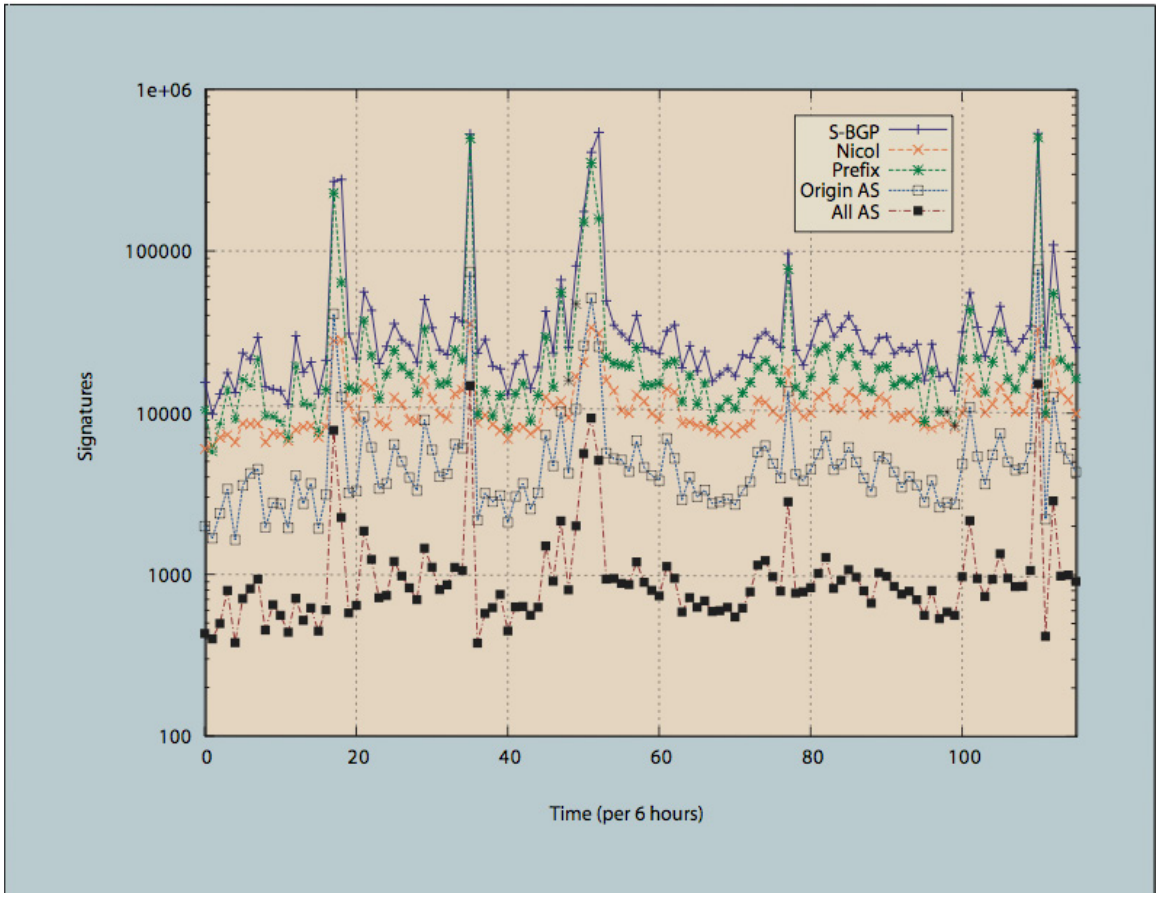


Figure 61. Number of signature validations required by scheme. Our schemes (Prefix, Origin AS, All AS) represent as much as a 97% decrease in validations over the S-BGP standard, thus making real-time path authentication possible due to the decrease in cryptographic computations required.

8.1.5 Compare all the attack profiles and define classes of attack profiles. Prepare and deliver a technical report on attack profiles. Prepare and submit journal/conference paper(s) using materials from this technical report.

Our literature survey provided comparisons of attacks against the BGP infrastructure from a variety of sources. The technical report we prepared is a comprehensive source for these attacks and potential solutions to these problems, as well as analysis as to the benefits and shortcomings of individual solutions [40]. This work has been expanded based on contributions from leading researchers in the field and will shortly be submitted to a major journal for consideration as a novel contribution to the research community, and particularly timely given the focus on BGP attacks.

8.1.6 Set up a testbed of the IC information infrastructure.

We used the AT&T network to generate a very large corpus of experimental data for use in evaluating potential cybersignal solutions to network attacks. This data represents tens of millions of individual flows, and has been extensively formatted and filtered for use in experimentation. We have aggregated the data to show almost 8.5 million individual sessions, and this has given tremendous insight into the nature of transactional signaling.

8.1.7 Simulate each attack profile on the testbed, and collect cyber signal data, including activity data, state change data, and performance impact data at candidate observable points throughout the cause-effect network of the attack profile.

We tested and reported on statistical models derived from the ASU efforts on differentiating cyber signal and noise. With the provided models and statistical software, we ran a series of tests over the breadth of experimental data on connection flows created from the AT&T research networks. The resulting information, based on multiple tests, was transferred to researchers at ASU for further analysis and evaluated for correlation with other results using cyber signal processing methods. Our own experiments, based on our data and cryptographic constructions we have derived, show that we can do real-time path authentication based on the computational savings our schemes provide. These schemes, if implemented by ASes, can prevent many of the attacks we outline against path modification and deletion. Please see Figure 61, for further detail. The results of this work have been submitted to a major networking conference, where we expect it to have significant impact in the field.

8.2 BGP Attack Classifications

Table 86 classifies some attacks on BGP.

Table 86. Classification of BGP Attacks

Origin Attacks

Attack Name	Threat	Agency	Action	Vulnerability	Target	State Effect	Performance Effect	Notes
Prefix hijacking - create a forged UPDATE claiming to be origin of some prefix	Any	Any	Forgery	Specification	Data	Availability, Integrity	Timeliness, Accuracy	This is the problem that origin authentication is really getting at.
ATOMIC_AGGREGATE modification - can cause deaggregation of prefixes	Any	Any	Spoof	Specification	Data	Availability	Accuracy	The ATOMIC_AGGREGATE field is set by routers to prevent deaggregation of routes. By allowing deaggregation, incorrect routing of more specific prefixes within the aggregate can result.

UPDATE eavesdrop - read the update from the UPDATE stream.	Any	Any	Read	Specification	Data	Confidentiality	None	This is a hard one to nail down. BGP UPDATES are generally considered public information (because they are flooded), but UPDATEs traversing private networks may be filtered or aggregated before being passed on.
Policy eavesdrop - read a policy in an UPDATE.	Any	Any	Read	Specification	Data	Confidentiality	None	BGP policy often is local to some community (hence the name community string), and is filtered in some cases. Exposure of this information will tell the adversary something about the organizations and relationships in the network.
Prefix Removal - remove a prefix advertisement from BGP UPDATE stream	Any	Any	Delete	Specification	Data	Availability	Timeliness, Accuracy	Cause prefix to be unavailable
Modifying Withdrawn Routes field in UPDATE	Any	Any	Spoof	Specification	System, Data	Availability, Integrity, Confidentiality	Accuracy, Precision	By modifying with Withdrawn Routes field, the attacker can eliminate legitimate routes from the routing table, and can repeatedly do so by replaying the attack.
Whack-a-mole ASes - create a bogus AS using a unused AS number.	Any	Any	Spoof	Specification	Data	Availability, Integrity	Timeliness, Accuracy	Spammers use these when nobody else will transit their traffic. These are particularly bad because they introduce a lot of noise into the global BGP update stream, and indirectly cause instability.
AS impersonation - claim to be an AS you are not.	Any	Any	Spoof	Specification	Data	Availability, Integrity	Timeliness, Accuracy, Precision	This is really a problem because you only need to convince one AS (out of the currently 16,000) that you are the claimed AS.

Path Attacks

Attack Name	Threat	Agency	Action	Vulnerability	Target	State Effect	Performance Effect	Notes
Path Removal - remove a path from BGP UPDATE stream	Any	Any	Delete	Specification	Data	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	May cause suboptimal or incorrect route to be selected. If used to mess with routing, then timeliness and accuracy are performance effects. If used to reroute toward controlled AS, could be used as confidentiality effect.
Policy Removal - remove a policy from BGP UPDATE message	Any	Any	Delete	Specification	Data	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	May cause suboptimal or incorrect route to be selected. If used to mess with routing, then timeliness and accuracy are performance effects. If used to reroute toward controlled AS, could be used as confidentiality effect.
UPDATE removal - remove an update message from the UPDATE stream	Any	Any	Delete	Specification	Data	Availability	Timeliness, Accuracy, Precision	Can cause the path, prefix, policy removal behavior. This can occur either at the BGP protocol or TCP layers.
Modify Path – add, remove, modify hops in the BGP path	Any	Any	Modify	Specification	Data	Availability, Confidentiality	Timeliness, Accuracy, Precision	May cause suboptimal or incorrect route to be selected. If used to mess with routing, then timeliness and accuracy are performance effects. If used to reroute toward controlled AS, could be used as confidentiality effect.

AS_PATH attribute modification - modify this field in the UPDATE message	Any	Any	Modify	Specification	Data	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	AS_PATH with an incorrect origin AS can play havoc with routing, causing blackholes. AS_PATH can be shortened, making the route appear more favourable to peers.
NEXT_HOP atribute modification - can cause routing changes	Any	Any	Modify	Specification	Data	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	Changing the NEXT_HOP in conjunction with path modification can cause an attacking router to control and engineer traffic patterns.
Modify Policy - change the policy such that the route becomes more or less desirable.	Any	Any	Modify	Specification	Data	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	May cause suboptimal or incorrect route to be selected. If used to mess with routing, then timeliness and accuracy are performance effects. If used to reroute toward controlled AS, could be used as confidentiality effect.
MULTI_EXIT_DISC modification can harm routing inside AS	Any	Any	Modify	Specification	Data	Availability	Timeliness, Accuracy	The multi exit discriminator (MED) is a way of determining which external link to progagate updates on, based on information from the peer. Modification of this can cause suboptimal routing within a peer AS.
LOCAL_PREF modification can harm routing inside AS	Any	Any	Modify	Specification	Data	Availability	Timeliness, Accuracy	The local preference is a metric that helps determine which external link to prefer for given prefixes. Manipulation of this value can cause suboptimal routing within the affected AS.
Path forgery - create a forged UPDATE with a bogus path for a known prefix.	Any	Any	(Forgery ?)	Specification	Data	Availability, Integrity	Timeliness, Accuracy	It really does not matter if the prefix is being advertised by some known AS. Whack-a-mole Ases (see below) are really good for creating a stream of these.
Modifying the NLRI field of the UPDATE message	Any	Any	Spoof	Specification	Data	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	By changing the network layer reachability information in the UPDATE message, routing can be disrupted through the system, since the actual routing advertisements can be forged.

Timing Attacks

Attack Name	Threat	Agency	Action	Vulnerability	Target	State Effect	Performance Effect	Notes
Forged OPEN message during BGP session	Any	Any	Termination	Specification	System	Availability	Timeliness, Accuracy	If the BGP speaker is in the Connect, Active or Established state, this message will force the connection to be closed, with the same effects as discussed above.
Bogus OPEN connection when router is waiting to establish connection	Any	Any	Spoof	Specification	System	Availability	Timeliness, Accuracy	If the router is in the OpenSent state, an OPEN message will cause the connection to be confirmed. When the real router sends an OPEN, the connection will be closed because of connection collision.
OPEN message arrives while OpenDelay timer in OpenSent state	Any	Any	Termination	Implementation	System	Availability	Timeliness, Accuracy	The router should not be in the OPEN_SENT state if the DelayOpen timer is sent, but an implementation error with the finite state machine can cause this. An attacker familiar with the implementaion could bring down the connection this way.
Sending KEEPALIVE when peering connection in Connect, Active or OpenSent	Any	Any	Termination	Specification	System	Availability	Timeliness, Accuracy	In any of these states, the BGP speaker moves into the Idle state and will not establish a connection with the intended peer.

state								
TCP SYN forgery	Any	Any	Spoof	Implementation	System	Availability, Integrity, Confidentiality	Timeliness, Accuracy, Precision	If the attacker sends a SYN to a BGP speaker, the real peer's SYN would look like a second connection. If the attacker keeps the connection alive by guessing the correct SYN ACK, a collision between the two connections could occur, dropping the legitimate connection as a result.
SYN flooding	Any	Any	Flood (Single Source)	Implementation	System	Availability	Timeliness, Accuracy	SYN floods are discussed in http://www.cert.org/advisories/CA-1996-21.html - by not responding to the SYN ACK, but opening a new TCP connection, the attacker can fill the buffer of available open connections to the router, preventing legitimate connections.
TCP SYN ACK hijacking	Any	Any	Spoof	Implementation	System	Availability, Integrity, Confidentiality	Timeliness, Accuracy	By responding to a SYN set up during a legitimate connection between two BGP peers, an attacker can send a SYN-ACK. If timed correctly, the legitimate peer's SYN-ACK will cause the TCP connection to be terminated, which brings down the BGP session in the process.
Altering BGP Timers	Any	Any	Spoof	Implementation	System	Availability, Integrity	Timeliness, Accuracy, Precision	Gaining control of the router could allow the attacker to modify the KeepAlive, Hold, or OpenDelay timers, causing peers to consider the connection unresponsive and terminate it.

Availability Attacks

Attack Name	Threat	Agency	Action	Vulnerability	Target	State Effect	Performance Effect	Notes
Route Flooding - flood a BGP speaker with more UPDATEs than it can handle.	Any	Any	Flood (Single Source)	Specification	System	Availability	Timeliness	This occurs naturally by table resets, and can be caused by forged TCP RST packets, or by forged BGP session termination messages.
Speaker death - shut down (process layer) or isolate (network wise) the BGP speak such that the BGP session closes.	Any	Any	Termination	Specification	System	Availability	Accuracy	This can be caused by forged TCP RST packets, or by forged BGP session termination messages. If the speaker comes back, this can cause flooding (both locally and globally).
Syntax error in message header - will close a BGP connection	Any	Any	Termination	Specification	System	Availability	Timeliness, Accuracy	Syntax errors cause the BGP speaker to close the connection and delete all routes associated with the connection, causing the router to reprocess information to determine how to now route those prefixes. This can cause a cascade effect with connected peers resetting their routes as well.
Syntax error in OPEN message will close connection	Any	Any	Termination	Specification	System	Availability	Timeliness, Accuracy	OPEN message syntax errors, such as errors in paramters or unsupported version numbers, will close a connection.

Receiving NOTIFICATION message brings down connection	Any	Any	Termination	Specification	System	Availability	Timeliness, Accuracy	Receiving NOTIFICATION message will cause BGP speaker to bring down the connection, and release and recalculate routes. This can cascade through to other routers.
Modifying Unfeasible Routes Length, Total Path Attribute Length attributes in UPDATE message	Any	Any	Termination	Specification	System, Data	Availability, Integrity	Timeliness, Accuracy	Modifying these parts of the UPDATE message will cause a NOTIFICATION message to be sent, terminating the connection.
Incorrect modification of Path Attributes can cause session failure	Any	Any	Termination	Specification	System, Data	Availability, Integrity	Timeliness, Accuracy	If the attributes are incorrectly modified, a parse error will occur, resulting in a NOTIFICATION message being sent and the connection being terminated.
Malformed UPDATE message will close connection	Any	Any	Termination	Specification	System	Availability	Timeliness, Accuracy	Sending an UPDATE message that contains errors will bring down the connection with the peer and cause all routes learned to be deleted and require recalculation. This can cascade to other routers.
TCP RST/FIN attack	Any	Any	Spoof	Implementation	System	Availability	Timeliness, Accuracy	Spoofing a TCP RST by guessing the correct sequence number will cause a TCP (and therefore BGP) connection to terminate. The attack works against the FIN as well, but there would be notification that the connection was closing.
Forcing manual reset of router	Any	Any	Termination	Implementation	System	Availability	Timeliness, Accuracy	Gaining control of the router through an attack like the SNMP buffer overflow exploit (eg. http://www.securityfocus.com/bid/1901) could allow the attacker to remotely shut down the router.
Link Cutting - hampering connectivity through making the network link inaccessible	Spies, Terrorists, Professional Criminals, Industrial Espionage	Any	Termination	Implementation	System	Availability	Timeliness, Accuracy	Described in http://www.research.att.com/~smb/papers/reroute.pdf, link cutting can take the form of the backhoe attack, ping of death or DoS of a given link. If the attacker knows the network topology, he or she can force packets to go through the paths they want through this attack.
Physical destruction of router	Spies, Terrorists, Professional Criminals	Human	Termination	Implementation	System	Availability	Timeliness, Accuracy	Physically disabling the router by destroying the interfaces or the machine itself is a possible attack. Physical sercurity of important network elements is always critical.
MD5 authentication attack	Any	Any	Authenti-cate	Implementation	System	Availability, Integrity, Confidentiality	Accuracy	While MD5 protection between peers can mitigate many of the above threats, attacking the authentication could yield ways to attack the protocol.Brute force and hash collision attacks are possible.

9. Conclusion

In this report, we consolidate our research/reporting for the duration of this project. We also include final project reports from our subcontractors. The first subcontractor, Symantec, was involved throughout the duration of the project. The second subcontractor, AT&T, was involved in the project until September 2004, at which time the PI for that subcontract left AT&T and the contract between ASU and AT&T terminated.

In addition to required reporting, we have produced several journal papers and one thesis on this research, which are referenced in each of the respective subsections of this final report. The final technical report and papers represent a summary of our major findings in as much as the scope of such research dissemination allows. We have exhibited the feasibility of our approach to cyber attack recognition, and additionally provided results to highlight the potential benefits of this work.

References

[1] N. Ye, “Mining computer and network security data.” in N. Ye (ed.), *The Handbook of Data Mining.* Mahwah, New Jersey: Lawrence Erlbaum Associates, 2003, pp. 617-636.

[2] P.E. Proctor. “Practical Intrusion Detection HandBook.” *Prentice Hall*, third edition, 2001.

[3] N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu, “Probabilistic techniques for intrusion detection based on computer audit data.” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 31, No. 4, 2001, pp. 266-274.

[4] G. Box, and A. Luceno, “Statistical Control by Monitoring and Feedback Adjustment.” *John Wiley & Sons*, New York, New York, 1997.

[5] Fisch EA, White GB (2000) Secure computers and networks: Analysis, design and implementation. CRC Press, Boca Raton.

[6] Ye N (2002) “QoS-centric stateful resource management in information systems.” Information Systems Frontiers, Vol. 4, No. 2, pp. 149-160.

[7] Roush M, Webb W (2000) Applied reliability engineering. University of Maryland, College Park, Maryland.

[8] Haines, J. W., Lippmann R. P., Fried D. J., Ziessman M. A., Tran E., Boswell S. B. "1999 DARPA Intrusion Detection Evaluation: Design and Procedures”, Technical Report 1062, February 15, 2001.

[9] John D. Howard, Thomas A. Longstaff, “A Common Language for Computer Security Incidents”, Technical Report SAND98-8667, Sandia National Laboratories, October 1998, <http://www.cert.org/research/taxonomy_988667.pdf>, (28 October 2003)

[10] Kristopher Kendall, “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems”, Massachusetts Institute of Technology, June 1999

[11] CERT Advisory “CA-2003-04 MS-SQL Server Worm”, 27 January 2003, <http://www.cert.org/advisories/CA-2003-04.html> (24 September 2003)

[12] CERT Advisory “CA-2000-04 Love Letter Worm”, 9 May 2000, <http://www.cert.org/advisories/CA-2000-04.html> (24 September 2003)

[13] Symantec security response, “W32.Sobig.F@mm”, 10 September 2003, <http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.f@mm.html> (24 September 2003)

[14] Symantec security response, “W32.HLLW.Fizzer@mm”, 30 May 2003, <http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.fizzer@mm.html> (24 September 2003)

[15] Symantec security response, “W32.Mimail.A@mm”, 01 August 2003, <http://securityresponse.symantec.com/avcenter/venc/data/w32.mimail.a@mm.html> (24 September 2003)

[16] Symantec security response, “W32.Bugbear@mm”, 11 July 2003, <http://securityresponse.symantec.com/avcenter/venc/data/w32.bugbear.b@mm.html> (24 September 2003)

[17] Symantec security response, “W32.Welchia.Worm”, 10 September 2003, <http://www.symantec.com/avcenter/venc/data/w32.welchia.worm.html#technicaldetails> (24 September 2003)

[18] Cox, M. J. (1998). Overview of security vulnerabilities in Apache httpd 1.3 (http://www.apacheweek.com/features/security-13).
[19] http://www.evicted.org/projects/writings/mftpadvisory.txt
[20] http://www.cert.org/advisories/CA-1998-01.html
[21] N. Ye and T. Farley, “Cyber Signal Detection: A New Approach to Intrusion Detection,” submitted.
[22] N. Ye and T. Farley, "A Signal-noise Separation Approach for Detecting Cyber Attacks," submitted.
[23] N. Ye, C. Newman and T. Farley, "A System-Fault-Risk Framework for Cyber Attack Classification and Profiling," Information Systems Frontiers, submitted.
[24] Bashettihalli Harish, Cyber Attack Profiling Using Cause Effect Networks, Thesis, Arizona State University, December 2004.
[25] T. C. Bailey, T. Sapatinas, K. J. Powell and W. J. Krzanowski. “Signal Detection in Underwater Sound Using Wavelets.” Journal of the American Statistical Association, Vol. 93, No.441, 1998, pp73-83.
[27] L. Atlas and P. Duhamel, “Recent developments in the core of digital signal processing.” IEEE Signal Processing Magazine, 16(1), 1999, pp. 16-31.
[28] A.K. Jain, P. Duin and J. Mao, “Statistical Pattern Recognition: Review.” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, 2000, pp. 4-37.
[29] F. Botella, J. Rosa-Herranz, J. J. Giner, S. Molina and J. J. Galiana-Merino, “A real-time earthquake detector with prefiltering by wavelets.” Computers & Geosciences, Volume 29, Issue 7, August 2003, pp. 911-919.
[30] Regmon utility from Sysinternals: http://www.sysinternals.com/ntw2k/source/regmon.shtml
[31] Win dump utility: http://windump.polito.it/ http://cert.uni-stuttgart.de/archive/bugtraq/1998/09/msg00036.html,
[32] “EZPublish Forum Discloses Installation Path and Database Password to Remote Users”, URL: http://securitytracker.com/alerts/2003/Apr/1006578.html
[33] NMAP security scanner URL: http://www.insecure.org/nmap/
[34] http://www.windowsitpro.com/Article/ArticleID/39845/39845.html
[35] (http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/counters/counters2_nbvr.asp).
[36] Pearl, Judea. “Fusion, propagation, and structuring in belief networks”. Artificial Intelligence, 29(3):241-288, 1986.
[37] Dempster, A.P. “A generalization of Bayesian inference”. Journal of the Royal Statistical Society, Series B 30 205-247, 1968.
[38] Shafer, Glenn. A Mathematical Theory of Evidence. Princeton University Press, 1976.
[39] Shortliffe, E.H. Computer-based medical consultation: MYCIN. Elsevier, 1976.
[40] K. Butler, T. Farley and P. McDaniel, "A Survey of BGP Security Issues and Solutions", Technical Report TD-5UGJ33, AT&T Labs - Research, Florham Park, NJ, February 2004. (revised June 2004).

List of Acronyms

ACF	Autocorrelation Function
ANOVA	Analysis of Variance
AR	Autoregressive Model
ARDA	Advanced Research and Development Activity
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
ARP	Address Resolution Protocol
AS	Autonomous System
ASU	Arizona State University
BGP	Border Gateway Protocol
CPU	Central Processing Unit
CSR	Customer Service Representative
DB	Database
DFC	Data, Feature and Characteristic
DNS	Domain Name Service
DOG	Derivative of Gaussian
DOS	Denial of Service
DW	Data Warehouse
EWMA	Estimated Weighted Moving Average
FSM	Finite State Model
FTP	File Transfer Protocol
GLM	Generalized Linear Model
GLS	Generalized Least Square
GP	General Purpose
HSA	Human Security Administrator
I&W	Indications and Warning
IC	Intelligence Community
ICA	Independent Component Analysis
IMA	Integrated Moving Average
IP	Internet Protocol
KS	Kolmogorov-Smirnov
MC	Multiple Correlation Coefficient
MED	Multi Exit Discriminator
OLS	Least Square Model
OLTP	Online Transaction Processing
PC	Principle Component
PCA	Principal Component Analysis
PLS	Partial Least Square
SFR	System Fault Risk
SMB	Server Message Block
SPC	Statistical Process Control
SPRT	Sequential Probability Ratio Test
SQL	Structured Query Language